UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

MÉMOIRE D'ÉTUDIANT

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

CRT-570

DEUX NOUVEAUX ALGORITHMES EXACTS POUR LE PROBLÈME DES M VOYAGEURS DE COMMERCE

par

Kim Nga Trinh

Mémoire de maîtrise présenté au département d'Informatique et de Recherche opérationnelle, Université de Montréal et dirigé par Messieurs Gilbert Laporte et Yves Nobert.

Cette étude a été publiée grâce à une subvention du fonds F.C.A.R. pour l'aide et le soutien à la recherche.

Université de Montréal
Centre de recherche sur les transports - Publication #570
Décembre 1987

Dépôt légal - 2e trimestre 1988

RÉSUMÉ

Ce mémoire contient la description de deux nouveaux algorithmes exacts pour le problème des m voyageurs de commerce (m-PVC). Nous avons mis au point, pour la construction du premier algorithme, une nouvelle formulation du m-PVC. À l'aide de celle-ci nous avons élaboré un algorithme de "branch-and-bound" très efficace pour le m-PVC basé sur la résolution à chaque noeud de l'arbre d'un problème de flot à coût minimum. Le deuxième algorithme que nous avons construit est une adaptation au cas du m-PVC d'un algorithme déjà connu pour le problème du voyageur de commerce. Ce deuxième algorithme utilise le problème d'affectation comme relaxation à chaque noeud de l'arbre d'énumération. Cet algorithme s'est avéré encore plus performant que le premier.

Mots-clés : problème de m voyageurs de commerce; problèmes de flot; problème d'affectation; branch-and-bound.

TABLE DES MATIÈRES

Page

IDENTIFICATION DU JURY. ii

SOMMAIRE. iii

TABLE DES MATIÈRES. iv

LISTE DES TABLEAUX. v

LISTE DES FIGURES vi

CHAPITRE 1 : INTRODUCTION 1

1.1 - Définition du problème 2

1.2 - Notation 3

CHAPITRE 2 : LE PROBLÈME DES m VOYAGEURS DE COMMERCE (m-PVC). 4

2.1 - Formulation du m-PVC 5

2.2 - Revue de la littérature sur les algorithmes pour le m-PVC. 8

CHAPITRE 3 : UN ALGORITHME EXACT POUR LE m-PVC BASÉ SUR UN PROBLEME DE FLOT A COUT MINIMUM 10

3.1 - Un problème de flot à coût minimum associé à (P1) 11

3.2 - Résolution de (P2) comme un problème de flot à coût minimum 14

3.3 - Construction d'heuristiques pour le m-PVC. 16

3.4 - Un algorithme de "branch-and-bound" pour le m-PVC. . . 17

3.5 - Résultats des tests sur ordinateur 25

CHAPITRE 4 : UN ALGORITHME EXACT BASÉ SUR LE PROBLÈME D'AFFECTATION . 29

4.1 - Modèle du PVC asymétrique. 30

4.2 - Transformation du m-PVC en un PVC. 31

4.3 - Algorithme de "branch-and-bound" basé sur le problème d'affectation 34

4.3.1 - Résolution de (P4) comme un problème d'affectation 34

4.3.2 - Algorithme de "branch-and-bound" 35

4.4 - Résultats des tests 35

CONCLUSION. 40

BIBLIOGRAPHIE 42

ANNEXE vii

REMERCIEMENTS xxxv

LISTE DES TABLEAUX

Page

Tableau 3.1 : Résultats des tests sur ordinateur. 26

Tableau 4.2 : Résultats des tests quand m est fixé. 36

Tableau 4.3 : Résultats des tests quand m est libre avec $\bar{m}$=5. 37

LISTE DES FIGURES

Page

Figure 2.1 : Solution réalisable du m-PVC avec n=5 et m=2 6

Figure 2.2 : Solution irréalisable du m-PVC avec n=5 et m=2 7

Figure 3.1 : Le problème de flot à coût minimum associé à (P2) sans (3.8). 14

Figure 4.1 : Matrice des distances du PVC associée à un m-PVC 32

Figure 4.2 : Correspondance entre un 2-tour et un tour par la transformation de Lenstra et al. 34

CHAPITRE 1

INTRODUCTION

1.1 Définition du problème

Le problème des m voyageurs de commerce (m-PVC) est celui rencontré par m voyageurs qui doivent visiter n villes puis retourner à leur point de départ, tout en minimisant leur distance totale parcourue. Chaque ville est visitée une seule fois par un seul voyageur. Le point de départ et de retour de ceux-ci est unique et nous le dénommerons par le dépôt.

En termes de la théorie des graphes, ce problème est la recherche de m cycles (circuits) de longueur totale minimale dans un graphe non orienté (orienté) tels que :

1- Le dépôt est contenu exactement une fois dans chacun de ces m cycles (circuits);

2- Chaque sommet différent du dépôt est contenu dans exactement un de ces m cycles (circuits).

Nous considérons dans cette thèse deux instances du m-PVC :

1- Le nombre de commis à utiliser est fixé à l'avance [1,13,27] à une certaine constante $\bar{m}$ où $1 < \bar{m} < n-1$. Dans ce cas, on utilise la notation $\bar{m}$-PVC.

2- Le nombre de commis à utiliser est libre [3,18], mais borné inférieurement et supérieurement par des constantes définies à l'avance notées respectivement $\underline{m}$ et $\bar{m}$. On peut donc écrire $1 < \underline{m} < m < \bar{m} < n-1$. Dans ce cas, nous utiliserons la notation m_{ℓ} - PVC.

Les applications du m-PVC sont nombreuses. Citons en particulier deux applications décrites par Lenstra et Rinnooy Kan dans [21] :

1- Construction de trajets pour une équipe d'hommes ("commis") devant visiter des boîtes téléphoniques ("villes") dans le nord de la Hollande;

2- Calcul du nombre minimum de véhicules ("commis") pouvant visiter 200 boîtes postales ("villes") dans la ville d'Utrecht.

1.2 Notation

Soit G = (N,A,C) le graphe orienté associé au m-PVC où N = {1,...,n} est un ensemble de noeuds représentant les n villes.

Dans cette étude, nous identifierons le dépôt à la ville 1 et les villes à visiter seront dénotées par 2,...,n.

A = {(i,j) : i,j ε N et i≠j} est un ensemble d'arcs représentant le trajet direct de la ville i vers la ville j pour toute paire de villes i et j.

$C_{nxn} = (c_{ij})$ est la matrice des distances associées aux arcs. Nous supposerons que c_{ij} est indépendant du commis particulier qui effectue le trajet de la ville i à la ville j. Nous supposons aussi que les véhicules sont identiques et chaque commis sera identifié à un véhicule donné.

Nous considérons ici le problème du m-PVC où C est asymétrique, c'est-à-dire que la distance séparant deux villes dépend du sens servi pour passer de l'une à l'autre.

CHAPITRE 2

LE PROBLÈME DES m VOYAGEURS DE COMMERCE (m-PVC)

2.1 Formulation du m-PVC

Soient X_{ij} des variables binaires telles que :

$$X_{ij} = \begin{cases} 1 & \text{si un des m commis passe de la ville i à la ville j} \\ 0 & \text{sinon} \end{cases}$$

Dans le reste du rapport, nous supposerons toujours que $i \neq j$ dans les notations c_{ij}, X_{ij}.

Le problème de m-PVC peut être formulé comme suit :

$$\text{Minimiser} \quad Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} X_{ij} \tag{2.1}$$

assujettie à :

(P1)

$$\sum_{j=2}^{n} X_{1j} = \sum_{i=2}^{n} X_{i1} = m \tag{2.2}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad (j=2,\ldots,n) \tag{2.3}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad (i=2,\ldots,n) \tag{2.4}$$

$$\sum_{\substack{i \in S \\ j \in S}} X_{ij} \leq |S| - 1 \qquad (S \subset N - \{1\}) \tag{2.5}$$

$$X_{ij} = 0, 1 \qquad (i,j=1,\ldots,n) \tag{2.6}$$

$$\underline{\dot{m}} \leq m \leq \bar{m} \tag{2.7}$$

L'équation (2.1) de (P1) évalue la distance totale parcourue par les m commis-voyageurs.

Les contraintes (2.2) jusqu'à (2.4) constituent les contraintes de degré pour le m-PVC : elles contrôlent le nombre de mouvements d'entrées et de sorties à chaque ville.

Les contraintes (2.5) sont appelées les contraintes d'élimination des sous-tours. Elles empêchent la formation de sous-tours illégaux qu'on pourrait obtenir en incluant dans (P1) seulement les contraintes de degrés. Notons que $|S|$ dénote la cardinalité du sous-ensemble S. La forme de ces contraintes est une généralisation directe des contraintes de Dantzig et al. [8] au cas du m-PVC. Contrairement au PVC, on se restreint ici aux sous-ensembles S de $\{2,\ldots,n\}$ car dans le m-PVC, les sous-tours contenant le dépôt sont permis. Les figures suivantes illustrent une solution réalisable et une solution irréalisable pour le m-PVC obtenues lorsque dans (P1) on relaxe les contraintes (2.5) :

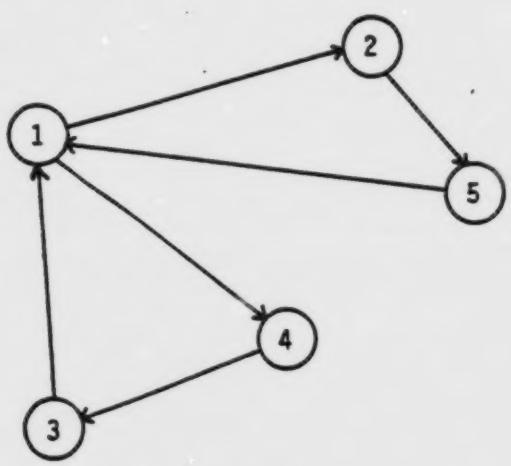


Figure 2.1 : Solution réalisable du m-PVC avec n=5 et m=2

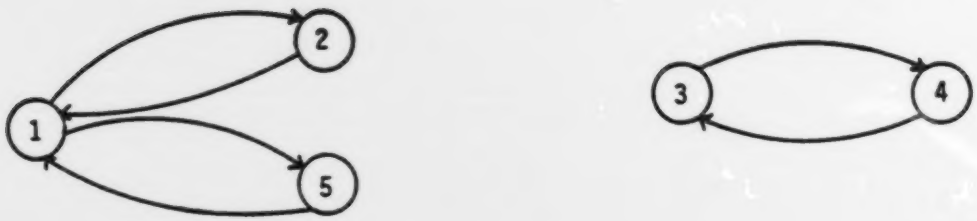


Figure 2.2 : Solution irréalisable du m-PVC avec n=5 et m=2

Les arcs (i,j) inclus dans les figures 2.1 et 2.2 correspondent aux variables X_{ij}= 1 dans la solution de (P1). Dans la figure 2.2, la contrainte suivante de (2.5) n'est pas satisfaite :

$$X_{34} + X_{43} = 2 > 1 = |S| - 1 \quad \text{avec } S = \{3,4\}$$

Les contraintes (2.6) et (2.7) sont appelées les contraintes d'intégrité sur les variables. Elles correspondent à la définition de celles-ci.

Les variables X_{ii} (i=1,...,n) doivent être exclues explicitement ou implicitement de toute solution de (P1), car elles n'ont aucun sens dans le contexte du problème. On leur associe une distance c_{ii} (i=1,...,n) tellement grand qu'elles seront absentes de toute solution optimale.

Notons que dans (P1), si le nombre de commis est fixé alors m est une constante donnée et (2.7) est inutile. Si le nombre de commis est libre alors m est une variable dans (2.2) et on suppose $\underline{m} \neq \bar{m}$.

Il existe d'autres modèles pour le m-PVC dont ceux de Gavish et Graves [12] pour le m-PVC symétrique et de Svestka et Huckfeldt [27] pour le m-PVC asymétrique.

2.2 Revue de la littérature sur les algorithmes pour le m-PVC

La forme générale des modèles pour le m-PVC est la suivante :

Min z = objectif linéaire

asujetti à :

(P.G.) Contraintes de degré
Contraintes d'élimination de sous-tours illégaux
Contraintes d'intégrité sur les variables

L'objectif à minimiser est une fonction linéaire des distances parcourues par les m-voyageurs. Le nombre de contraintes de degré sous la forme (2.3) et (2.4) est égal à 2(n-1). Le nombre de contraintes d'élimination de sous-tours illégaux décrites en 2.1 est égal à :

$$2\left[\binom{n-1}{2} + \ldots + \binom{n-1}{\frac{1}{2}(n-3)}\right] + \binom{n-1}{\frac{1}{2}(n-1)} + \binom{n-1}{n-2} \quad \text{si } n \text{ impair}$$

ou égal à :

$$2\left[\binom{n-1}{2} + \ldots + \binom{n-1}{\frac{1}{2}(n-2)}\right] + \binom{n-1}{n-2} \quad \text{si } n \text{ pair .}$$

On remarque que ce nombre est exponentiel en n. De plus, les contraintes d'intégrité sur les variables de (P.G.) ajoutent la touche finale à son extrême complexité.

Les algorithmes décrits dans la littérature, y compris les nôtres, sont basés sur le choix d'une relaxation de (P.G.).

Certains auteurs relaxent de façon lagrangienne les contraintes de degré aux noeuds différents du dépôt dans (P.G.) [25]. Ali et Kernington [1] et Gavish et Srikantn [13] ont construit des algorithmes qui sont des

extensions directes des travaux de Held et Karp [15,16] au PVC. Ces algorithmes sont basés sur la relaxation lagrangienne. La valeur des variables duales est calculée par une technique de sous-gradient due à Sandi [24] pour ces algorithmes.

D'autres auteurs relaxent les contraintes d'élimination de sous-tours illégaux et conservent les contraintes de degré; ce qui conduit à un problème de flot à coût minimum (cas asymétrique) ou à un problème de "b-matching" [9] (cas symétrique). Svestka et Huckfeldt [27] ont généralisé les travaux de Bellemore et Malone [4] sur le PVC. Ils utilisent une méthode de "branch-and-bound" où on tient compte des contraintes d'élimination de sous-tours implicitement au niveau du branchement.

Laporte et Nobert [1] ainsi que Nobert [23] ont utilisé une relaxation complète initialement des contraintes d'élimination de sous-tours illégaux et des contraintes d'intégrité sur les variables. Ils utilisent une méthode de "branch-and-bound" où l'intégrité des variables est maintenue à l'aide de coupes de Gomory ou au niveau du branchement.

Le principe général sur lequel est basé notre premier algorithme pour le m-PVC consiste en une relaxation des contraintes d'élimination de sous-tours illégaux et conservent les contraintes de degré, ce qui conduit à un problème de flot à coût minimum. Le deuxième algorithme que nous avons construit est une adaptation au cas du m-PVC d'un algorithme déjà connu pour le problème du voyageur de commerce. Ce deuxième algorithme utilise le problème d'affectation comme relaxation du PVC (c'est-à-dire sans les contraintes d'élimination de sous-tours illégaux) à chaque noeud de l'arbre d'énumération.

CHAPITRE 3

UN ALGORITHME EXACT POUR LE m-PVC BASÉ SUR UN PROBLÈME DE FLOT À COÛT MINIMUM

3.1 Un problème de flot à coût minimum associé à (P1)

Nous voulons faire remarquer d'abord que dans (P1) il est possible d'éliminer les variables X_{i1} et X_{1j} car :

$$\sum_{j=2}^{n} X_{ij} = \begin{cases} 0 \iff X_{i1} = 1 \\ 1 \iff X_{i1} = 0 \end{cases} \quad \text{pour } i \neq 1 \qquad (3.1)$$

et

$$\sum_{i=2}^{n} X_{ij} = \begin{cases} 0 \iff X_{1j} = 1 \\ 1 \iff X_{1j} = 0 \end{cases} \quad \text{pour } j \neq 1 \qquad (3.2)$$

Les équivalences sont des conséquences directes des contraintes de degré. En particulier, si $X_{ij} = 1$ ($i \neq 1$, $j \neq 1$) alors $X_{i1} = 0$ et $X_{1j} = 0$. De plus, le nombre total d'arcs (i,j) utilisés ($X_{ij} = 1$) dans toute solution réalisable est :

$$\sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} = n + m - 1 \qquad (3.3)$$

En combinant (2.2), (2.7) et (3.3) on obtient facilement :

$$n - \bar{m} - 1 \leq \sum_{i=2}^{n} \sum_{j=2}^{n} X_{ij} \leq n - \underline{m} - 1 \qquad (3.4)$$

En tenant compte des remarques précédentes, on obtient que (P1) est équivalent au modèle (P2) suivant :

$$\text{Minimiser} \quad Z = D - \sum_{i=2}^{n} \sum_{j=2}^{n} (c_{i1} + c_{1j} - c_{ij}) X_{ij} \qquad (3.5)$$

<u>assujettie à</u> :

$$\sum_{j=2}^{n} X_{ij} \leq 1 \qquad (i \in N - \{1\}) \qquad (3.6)$$

$$\sum_{i=2}^{n} X_{ij} \leq 1 \qquad (j \in N - \{1\}) \qquad (3.7)$$

(P2)

$$\sum_{i,j \in S} X_{ij} \leq |S| - 1 \qquad (S \subseteq N - \{1\}) \qquad (3.8)$$

$$n - \bar{m} - 1 \leq \sum_{i=2}^{n} \sum_{j=2}^{n} X_{ij} \leq n - \underline{m} - 1 \qquad (3.9)$$

$$X_{ij} = 0, 1 \qquad (i,j \in N - \{1\}) \qquad (3.10)$$

où $$D = \sum_{i=2}^{n} c_{i1} + \sum_{j=2}^{n} c_{1j} \qquad (3.11)$$

En posant $$S_{ij} = (c_{1i} + c_{i1} + c_{1j} + c_{j1}) - (c_{1i} + c_{ij} + c_{j1})$$

on obtient : $$S_{ij} = c_{i1} + c_{1j} - c_{ij} \qquad (3.12)$$

On peut alors écrire :

$$Z = D - \sum_{i=2}^{n} \sum_{j=2}^{n} S_{ij} X_{ij} \qquad (3.13)$$

La quantité S_{ij} représente le gain obtenu en remplaçant les aller-retour $1 \to i \to 1$ et $1 \to j \to 1$ par le tour $1 \to i \to j \to 1$.

S_{ij} correspond de plus au gain de l'algorithme de Clarke et Wright [7] pour le problème d'ordonnancement de véhicules.

La constante D dans la fonction objectif correspond à la distance totale parcourue si chaque ville de N - {1} est servie par un aller-retour. Dans l'équation (3.13) on remarque que D est une constante et, donc, il n'est pas nécessaire d'en tenir compte dans la fonction objectif. En multipliant par -1, l'objectif de (P2) devient :

$$\text{Maximiser} \quad Z = \sum_{i=2}^{n} \sum_{j=2}^{n} S_{ij} X_{ij} \qquad (3.15)$$

Il s'agit donc dans (P2) de trouver la meilleure correction à la solution initiale où chaque ville est servie par un seul commis.

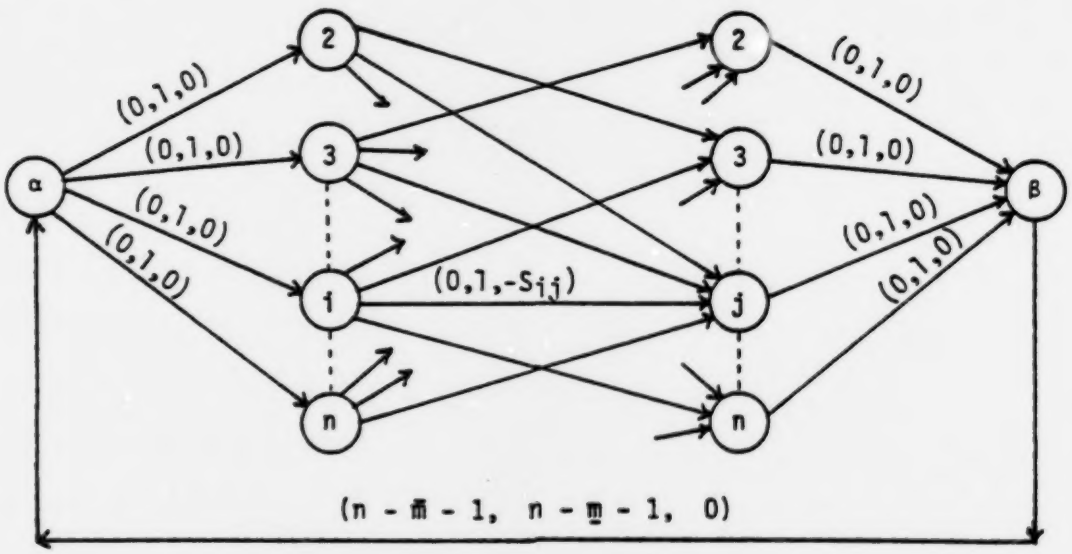
Le modèle (P2) que nous venons d'élaborer est à notre connaissance nouveau. Son principal intérêt réside dans le fait qu'en relaxant les contraintes d'élimination des sous-tours illégaux on obtient un problème noté (P3) qui peut se résoudre comme un problème de flot à coût minimum. Nous donnons des explications détaillées de ce fait à la section 3.2. Puisque le problème de flot à coût minimum est bien résolu en recherche opérationnelle et aussi puisque la borne inférieure qu'il nous fournit est excellente (comme on le verra), son utilisation dans un algorithme de "branch-and-bound" pour résoudre le m-PVC de façon exacte doit en théorie donner de bons résultats. Notons de plus que l'utilisation de (P2) comme relaxation n'implique pas l'ajout de villes artificielles comme c'est le cas de la relaxation utilisée au chapitre 4.

La section 3.3 décrit la construction d'heuristiques pour le m-PVC. Un algorithme exact de type "branch-and-bound" pour le m-PVC en utilisant la relaxation des contraintes d'élimination des sous-tours illégaux se

trouve dans la section 3.4. La section 3.5 contient des résultats de tests sur ordinateur concernant l'utilisation de cet algorithme.

3.2 Résolution de (P2) comme un problème de flot à coût minimum

Le problème (P2), sans (3.8), peut être résolu comme un problème de flot à coût minimum; noté (P3), dans le réseau suivant :



Légende : (borne inférieure, borne supérieure, coût)

Figure 3.1 : Le problème de flot à coût minimum associé à (P2) sans (3.8)

Dans la figure 3.1, chaque ville $i \in N - \{1\}$ est représentée par deux noeuds. Une source α et un puits β sont aussi ajoutés. Dans ce qui suit, on explique pourquoi (P2) sans (3.8) est équivalent à (P3). Les arcs de la figure 3.1 peuvent être divisés en quatre catégories :

(i) arcs (i,j) où $i,j \in N - \{1\}$:

Dans la solution optimale de (P3), si la valeur du flot sur un arc (i,j) est égale à 1 alors un des commis exécute le trajet de la ville i vers la ville j.

(ii) arcs (α,i) où $i \in N - \{1\}$:

A cause de la conservation du flot, on a $X_{\alpha i} = \sum_{j=2}^{n} X_{ij}$. Puisqu'on a imposé $0 \leq X_{\alpha i} \leq 1$, alors la contrainte (3.6) est satisfaite pour le noeud i. De plus, par l'équivalence (3.1) on a $X_{\alpha i} = 1 - X_{i1}$, i.e. $X_{\alpha i} = 0 \iff X_{i1} = 1$ et d'où un passage de la ville i à la ville 1 est effectué par l'un des m commis.

(iii) arcs (j,β) où $j \in N - \{1\}$:

Encore ici, la conservation du flot implique $X_{j\beta} = \sum_{i=2}^{n} X_{ij}$. Puisque $0 \leq X_{j\beta} \leq 1$, alors la contrainte (3.7) est satisfaite pour le noeud j. De plus, par l'équivalence (3.2) on a $X_{j\beta} = 1 - X_{1j}$, i.e. $X_{j\beta} = 0 \iff X_{1j} = 1$. Un passage de la ville 1 à la ville j est alors effectué par l'un des m commis.

(iv) arc (β,α) :

Dans le réseau de la figure 3.1, la valeur du flot sur l'arc (β,α) est $X_{\beta\alpha} = \sum_{i=2}^{n} \sum_{j=2}^{n} X_{ij}$. Puisque $n - \bar{m} - 1 \leq X_{\beta\alpha} \leq n - \underline{m} - 1$, la

contrainte (3.9) est respectée. La valeur de m correspondant à la solution optimale dans le réseau est égale à $n - X_{\beta\alpha} - 1$.

Finalement, le coût total d'une solution réalisable de (P3) est égal à $S = \sum_{i=2}^{n} \sum_{j=2}^{n} S_{ij} X_{ij}$. En dénotant par S* la valeur de S pour la solution optimale de (P3), on obtient par (3.13) que $Z^* = D - S^*$ où Z^* est la valeur optimale de (P2) sans (3.8).

3.3 Construction d'heuristiques pour le m-PVC

Nous avons construit deux méthodes heuristiques pour le m-PVC. La première des deux méthodes est basée sur une transformation du m-PVC en un PVC. Un exemple de ce type de transformation est donné à la section 4.1 de ce texte. Par la suite, l'heuristique de Bodin pour le PVC asymétrique décrit dans [5] a été utilisé pour trouver une solution réalisable (un tour) et on améliore ce dernier à l'aide d'un heuristique 2-OPT [22]. Les résultats obtenus furent décevants et nous avons construit l'heuristique suivant.

Supposons que le problème (P3) ait été résolu et que la solution optimale contienne des sous-tours illégaux. Plus précisément, on supposera que cette solution puisse être décrite par m sous-graphes disjoints $G_1(N_1,A_1,C_1),\ldots, G_m(N_m,A_m,C_m)$ où $1 \in N$, et où $A_2,\ldots,A_m$ forment des sous-tours illégaux. Le rationnel sous-jacent à cet heuristique est celui de la fusion des sous-tours illégaux entre eux ou avec des sous-tours reliés au dépôt de façon à les éliminer. À l'aide d'un calcul simple de pénalités, on tentera de minimiser le plus possible l'augmentation dans la longueur totale des sous-tours lors des fusions successives.

Supposons que le nombre de commis soit fixé à $\bar{m}$ et qu'on veut relier le sous-tour A_Γ au dépôt. La pénalité minimale encourue sera

$$p_\Gamma = \min_{(i,j)\in A_1} \min_{(k,\ell)\in A_\Gamma} \{c_{i\ell} + c_{kj} - c_{ij} - c_{k\ell}\} \qquad (3.16)$$

Le sous-tour $A_{\Gamma\star}$ qui sera relié au dépôt est celui pour lequel p_Γ est minimal. Après chaque fusion, les pénalités sont réévaluées si nécessaire et un nouveau sous-tour illégal est relié au dépôt.

Lorsque le nombre de commis est libre, une version un peu différente de cet heuristique a été construite pour permettre la fusion de deux sous-tours illégaux entre eux.

On retrouvera à la section 3.5 des résultats de tests sur ordinateur comparant les solutions fournies par cet heuristique aux solutions optimales obtenues par la méthode exacte décrite en 3.4.

3.4 Un algorithme de "branch-and-bound" pour le m-PVC

Nous décrivons ici un nouvel algorithme de "branch-and-bound" pour résoudre le m-PVC. La relaxation (P3) de (P2) est tout d'abord résolue par l'algorithme RNET [14], calculant une circulation à coût minimal dans un réseau. Si la solution obtenue ne contient pas de sous-tours illégaux, alors la solution est réalisable et aussi optimale pour (P2). Sinon, les sous-tours illégaux seront éliminés graduellement par la fixation de variables X_{ij} à la valeur 0 ou 1 dans l'arbre d'énumération.

La méthode de branchement utilisée est celle de Garfinkel [11], modifiée par Carpaneto et Toth [6]. Nous la décrirons en détail plus loin. À chaque noeud de l'arbre d'énumération correspond donc un problème

de flot (PF) dont le domaine réalisable est un sous-ensemble du domaine réalisable du problème de flot (PF) original associé au m-PVC. La racine de l'arbre correspond justement à ce dernier problème de flot. Supposons que l'on se trouve à un noeud quelconque de l'arbre, noté P, et qu'on ait résolu le (PF) associé au domaine réalisable F(P) ou tout simplement qu'on ait calculé une borne inférieure sur la solution optimale du (PF) associé à (P).

Dénotons par $\underline{Z}$ le coût de cette solution. De plus, $\bar{Z}$ signifiera le coût du meilleur m-tour en main. Trois situations peuvent se produire :

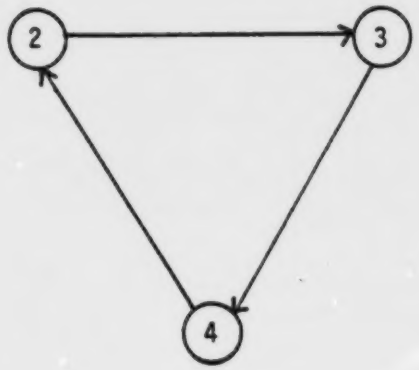
a) $\underline{Z} > \bar{Z}$ (au début, si aucun m-tour n'est connu, on pose $\bar{Z} = \infty$);

b) $\underline{Z} < \bar{Z}$, mais la solution contient des sous-tours illégaux;

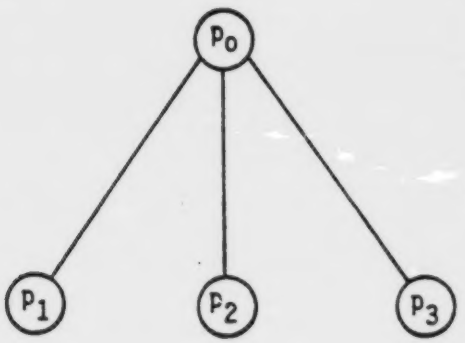
c) $\underline{Z} < \bar{Z}$ et la solution ne contient pas de sous-tours.

Dans la situation a), F(P) ne peut contenir la solution optimale du m-PVC et il ne sert à rien d'analyser plus à fond le noeud P et ce noeud est éliminé. Il faut choisir un autre noeud dans l'arbre d'énumération à analyser. La façon dont on choisira ce nouveau noeud sera appelée méthode de branchement. Il y a plusieurs méthodes de branchement possibles. Ici, nous utilisons la méthode de branchement de Carpaneto et Toth [6] qui s'est avérée très efficace pour le m-PVC. Cette méthode est basée sur la sélection du noeud pendant possédant la plus petite borne inférieure parmi les noeuds pendants au niveau le plus élevé. Considérons l'exemple suivant : supposons qu'on a résolu le (PF) original associé au m-PVC et que la solution obtenue contient des sous-tours (sinon on possède déjà la solution optimale du m-PVC). On choisit d'éliminer le sous-tour qui a le plus petit nombre d'arcs qui ne sont pas fixés à 1 et on crée les noeuds $p_1, p_2, \ldots, p_k$

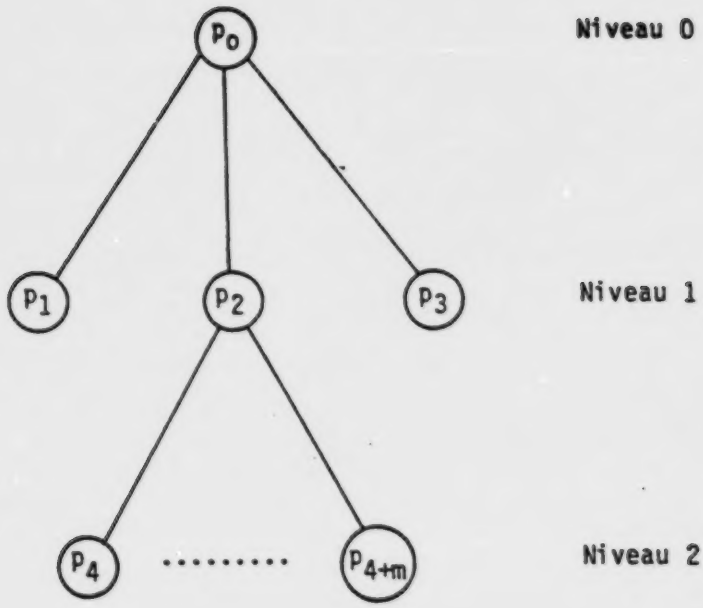
en fixant pour chacun certaines variables à 0 ou à 1. On solutionne ensuite les (PF) associés aux noeuds p_i. Supposons que le sous-tour choisi soit :



L'arbre d'énumération implicite possèdera présentement cet aspect :



Quel est le prochain noeud p_i que nous allons considérer. Selon notre critère de branchement, c'est celui qui possède la solution optimale de moindre coût. Supposons qu'il s'agisse de p_2. Supposons que la solution du (PF) associé à (P2) contienne elle aussi des sous-tours illégaux. Supposons que celui ayant le plus petit nombre d'arcs non imposés à 1 contient m arcs libres. On crée de nouveaux noeuds à partir de p_2 notés $p_4, \ldots, p_{4+m}$ pour lesquels on résoud aussi le (PF) associé. L'arbre d'énumération implicite a maintenant l'allure suivante :



Le prochain noeud à considérer dans un tel arbre sera sélectionné selon le même critère que précédemment parmi P_1, P_3, P_4,..., P_{4+m} .

On continue ainsi jusqu'à ce qu'aucun noeud de l'arbre n'ait besoin d'être analysé.

D'autres méthodes de branchement ont été analysées dans la littérature. En particulier, il est possible aussi d'utiliser une méthode LIFO (last-in-first-out) où le noeud pendant choisi est toujours un noeud fils du dernier noeud analysé. L'avantage de la méthode de branchement utilisée ici vient surtout du fait que la taille de l'arbre sera probablement minimisée car lorsqu'on trouve une solution réalisable au m-PVC, celle-ci aura un $\bar{Z}$ probablement plus petit qu'une méthode LIFO. Ainsi, plusieurs noeuds seront par la suite mis hors considération sans les subdiviser en sous-problèmes.

Nous développons maintenant, étape par étape, une description formelle de l'algorithme. Définissons d'abord :

$\bar{Z}$: la valeur de la meilleure solution réalisable trouvée;

Z^* : la valeur de la solution optimale de (P2) ($\bar{Z} > Z^*$);

Z_h : la valeur de la solution optimale du (PF) au noeud h de l'arbre d'énumération;

I_h : l'ensemble des arcs fixés à la valeur 1 au noeud h de l'arbre d'énumération;
(i.e. $I_h = \{(i,j) \mid X_{ij} = 1$ au noeud h$\}$);

E_h : l'ensemble des arcs fixés à la valeur 0 au noeud h de l'arbre d'énumération;
(i.e. $E_h = \{(i,j) \mid X_{ij} = 0$ au noeud h$\}$).

Étape 0 : (Initialisation)

On calcule $D = \sum_{i=2}^{n} C_{i1} + \sum_{j=2}^{n} C_{1j}$. On pose $I_1 = E_1 = \emptyset$ et $\bar{Z} = \infty$. Le problème (P3) est résolu par RNET. Soit Z_1 la valeur du (PF) original. ($Z_1 = D - S_1$). Si la solution ne contient pas de sous-tours illégaux, alors l'algorithme se termine et $Z^* = Z_1$, on a trouvé la solution optimale pour (P2), sinon on calcule $\bar{Z}$ d'après l'étape 5. On insère le noeud 1 de l'arbre d'énumération dans la queue.

Étape 1 : (sélection d'un noeud)

Si la queue est vide, alors l'algorithme se termine et $Z^* = \bar{Z}$, sinon le noeud pendant h dans l'arbre d'énumération ayant la plus petite valeur Z_h est choisi.

Étape 2 : (partitionnement)

La solution du noeud h sélectionné contient des sous-tours illégaux. On crée les noeuds "fils" du noeud h en choisissant le sous-tour avec le plus petit nombre d'arcs, noté L, non fixés à 1 (i.e. le plus petit nombre d'arcs n non contenus dans I_h) .

Soit $\{(i_1,j_1), (i_2,j_2), \ldots, (i_L,j_L)\}$ l'ensemble des arcs dans ce sous-tour qui ne sont pas dans I_h . On crée alors n noeuds "fils" du noeud h notés 1, ..., n par :

$$I_g = \begin{cases} I_h & (g=1) \\ I_h \cup \{(i_u,j_u) : u = 1,\ldots,g-1\} & (g=2,\ldots,L) \end{cases} \qquad (3.17)$$

$$E_g = E_h \cup \{(i_g,j_g)\} \qquad (g=1,\ldots,L) \qquad (3.18)$$

Pour chacun des noeuds fils g du noeud h, les étapes 3, 4, 5 sont exécutées.

Étape 3 : (optimisation du problème de flot)

Le problème de flot g contraint par I_g et E_g est résolu en utilisant RNET. Soit S_g le coût maximum associé et $Z_g = D - S_g$. Si $Z_g > \bar{Z}$ et $g < n$ alors on considère le prochain noeud fils (g=g+1) et on répète l'étape 3. Si $Z_g > \bar{Z}$ et $g = n$ alors on exécute l'étape 6.

Étape 4 : (tester la réalisabilité)

Si $Z_g < \bar{Z}$ et la solution courante ne contient pas de sous-tours illégaux alors on a trouvé une solution réalisable qui est meilleure que la dernière solution réalisable; on doit garder cette nouvelle solution et on pose $\bar{Z} = Z_g$.

Si $Z_g < \bar{Z}$ et la solution courante contient des sous-tours illégaux alors on complète les sous-tours par l'heuristique en exécutant l'étape 5. On insère le noeud g dans la queue.

S'il existe encore des noeuds fils à examiner alors on calcule g = g+1 et on répète l'étape 3 sinon on exécute l'étape 6.

Étape 5 : (fusion des sous-tours illégaux par l'heuristique)

La solution courante contient m sous-graphes disjoints : $G_1(N_1,A_1, C_1)$, ..., $G_m(N_m,A_m,C_m)$. Ces sous-graphes sont tels que $1 \in N_1$ et A_2, ..., A_m forment des sous-tours. Pour obtenir une solution réalisable, il suffit de fusionner successivement un sous-tour $G_r(1 \in \{2,...,m\})$ à G_1 en enlevant un arc (i,j) de A_1 et un arc (k,ℓ) de A_r et en introduisant deux nouveaux arcs (i,ℓ) et (k,j). Si m est libre on peut fusionner un sous-tour G_r avec un autre sous-tour $G_s(r,s\in\{2,...,m\}$ et $i\neq s)$ en enlevant un arc (i,j) de A_r et un arc (k,ℓ) de A_s et en introduisant deux nouveaux arcs (i,ℓ) et (k,j). Chaque fois, on diminue le nombre de sous-tours de 1. La procédure suivante est répétée jusqu'à ce que tous les sous-tours illégaux soient éliminés.

(i) Calculer les pénalités en liant les sous-tours: soit pt la pénalité obtenue si on relie un sous-tour au dépôt:

$$pt = \min_r \min_{(i,j)\in A_1} \min_{(k,\ell)\in A_r} \{c_{i\ell} + c_{kj} - c_{ij} - c_{k\ell}\} \qquad (3.19)$$

Soit ps la pénalité si deux sous-tours sont fusionnés :

$$ps = \min_r \min_{(i,j)\in A_r} \min_{(k,\ell)\in A_s} \{c_{i\ell} + c_{kj} - c_{ij} - c_{k\ell}\} \qquad (3.20)$$

où $s\neq r$

Soit p = min (pt,ps)

Si p = pt, soit r* et (i*,j*), (k*,ℓ*), le sous-tour et les arcs satisfaisant (3.18).

(ii) Fusionner G_{r*} avec G_1 :

$$N_1 = N_1 \cup N_{r*} \text{ et}$$

$$A_1 = A_1 \cup A_{r*} \cup \{(i*,\ell*),(k*,j*)\} - \{(i*,j*),(k*,\ell*)\}$$

$$\bar{Z}_g = Z_g - c_{i*j*} - c_{k*\ell*} + c_{i*\ell*} + c_{k*j*}$$

Si p = ps, soit r*, s* et (i*,j*), (k*,ℓ*), les sous-tours et les arcs satisfaisant à (3.19).

(ii) Fusionner G^*_r avec G^*_s

$$N_{s*} = N_{s*} \cup N_{r*} \text{ et}$$

$$A_{s*} = A_{s*} \cup A_{r*} \cup \{(i*,\ell*),(k*,j*)\} - \{(i*,j*),(k*,\ell*)\}$$

$$\bar{Z}_g = Z_g - c_{i*j*} - c_{k*\ell*} + c_{i*\ell*} + c_{k*j*}$$

(iii) Mettre à jour $\bar{Z}$ si $Z_g < \bar{Z}$

Étape 6 : (mise à jour de la queue)

On élimine tous les noeuds h de la queue tels que $Z_h > \bar{Z}$. On retourne ensuite à l'étape 1.

3.5 Résultats de tests sur ordinateur

Dans le tableau 3.1, on retrouve les résultats des tests effectués avec cet algorithme lorsque m est fixé à 2, 5, 10, 20, 40 pour n = 80, 100, 120, 140, 160 villes. Dans tous les cas, le graphe est complet.

Les c_{ij} sont générés aléatoirement dans l'intervalle [0,100] à l'aide d'une distribution uniforme. Ils sont tronqués à l'entier le plus près. Tous les problèmes sont résolus par l'ordinateur CYBER 173 de l'Université de Montréal en utilisant le compilateur FTN5. Le temps de résolution d'un problème est limité à 300 secondes. On remarquera que seulement un problème n'a pas été résolu avec ce temps limite.

La notation utilisée dans le tableau 3.1 est la suivante :

Z_H : valeur de la solution obtenue par l'heuristique à l'étape 0;

Z_0 : valeur de la solution optimale de (P3) à l'étape 0 (avec $I_1 = E_1 = 0$);

Z^* : valeur de la solution optimale de (P2).

Notons qu'on aura nécessairement $Z_r < Z^* < Z_H$. Les descriptions des variables dans le tableau 3.1 sont :

SUCCÈS : nombre de problèmes résolus avec succès (temps < 300 secondes) divisé par le nombre de problèmes testés;

NB-SSTOURS : nombre de problèmes contenant des sous-tours après la résolution du premier problème de flot à l'étape 0;

MOY-SSTOURS : nombre moyen des sous-tours contenu dans la solution initiale;

Tableau 3.1

Résultats de tests sur ordinateur

n	m	SUCCES	NB-SSTOURS	MOY-SSTOURS	$100(Z_H - Z_0)/Z_0$	$100(Z_H - Z^*)/Z^*$	$100(Z^* - Z_0)/Z^*$	NOEUDS	DESC	TEMPS
80	2	3/3	3	2.6	6.7	3.1	0.4	5.3	1.0	61.4
	5	3/3	2	1.0	2.9	3.2	0.2	2.6	1.0	19.4
	10	3/3	2	1.3	4.4	4.1	0.2	2.3	0.3	12.9
	20	3/3	2	1.0	1.9	1.7	0.1	2.3	0.3	10.7
	40	3/3	0	0.0	-	0.0	0.0	0.0	0.0	3.7
100	2	3/3	3	1.6	2.8	2.1	0.6	9.0	5.0	159.4
	5	3/3	2	0.6	1.9	1.4	0.4	1.0	0.0	30.5
	10	3/3	2	0.6	4.2	0.4	0.3	4.0	0.6	35.7
	20	3/3	2	1.0	1.8	1.8	0.0	1.0	0.0	10.7
	40	3/3	1	0.3	0.5	0.2	0.1	0.3	0.0	9.3
120	2	2/3	3	2.6	6.5	6.1	0.8	3.0	0.6	52.9
	5	3/3	3	2.0	4.7	4.4	0.4	2.3	0.3	37.9
	10	3/3	1	0.3	2.7	2.1	1.6	1.3	0.3	50.2
	20	3/3	3	1.0	0.9	0.6	0.3	1.0	0.0	27.1
	40	3/3	1	0.3	0.7	0.1	0.0	0.6	0.3	20.7
140	2	3/3	3	1.6	5.1	5.1	0.0	5.0	1.0	93.4
	5	3/3	2	3.0	5.5	3.5	1.2	5.0	0.3	131.1
	10	3/3	3	2.6	5.5	5.4	0.1	3.6	1.3	79.8
	20	3/3	3	1.0	1.2	1.1	0.1	1.0	0.0	29.6
	40	3/3	1	0.3	0.5	0.5	0.0	0.6	0.0	23.8
160	2	2/2	2	3.5	15.3	14.1	0.7	6.5	2.0	180.4
	5	2/2	2	2.5	8.9	8.4	0.3	4.0	1.0	98.4
	10	2/2	2	2.0	2.7	2.5	0.2	1.5	0.0	47.3
	20	2/2	1	1.0	1.5	1.5	0.0	2.0	0.0	50.5
	40	2/2	0	0.0	-	0.0	0.0	0.0	0.0	23.8

$100(Z_H - Z_0)/Z_0$: la différence relative moyenne entre la valeur de la solution réalisable obtenue par l'heuristique à l'étape 0 et la valeur optimale du premier problème de flot;

$100(Z_H - Z^*)/Z^*$: la différence relative moyenne entre la valeur optimale obtenue par l'heuristique à l'étape 0 et la valeur optimale du m-PVC;

$100(Z^* - Z_0)/Z^*$: la différence relative moyenne entre la valeur optimale du m-PVC et la valeur optimale du premier problème de flot;

NOEUDS : nombre moyen de noeuds dans l'arbre d'énumération;

DESC : nombre moyen des noeuds examinés dans l'arbre d'énumération (i.e. ceux avec descendants);

TEMPS : le temps moyen en secondes pour résoudre un problème.

On peut tirer les conclusions suivantes à partir du tableau 3.1.

(i) l'algorithme a bien fonctionné, presque tous les problèmes ont été résolus en un temps raisonnable. Le problème de plus grande taille contient 160 villes;

(ii) pour une valeur de m donnée, la difficulté de résolution du problème diminue quand m augmente (mesuré par le nombre de noeuds dans l'arbre d'énumération ou par le temps de résolution). Ce phénomène a aussi été observé dans [19];

(iii) la relaxation utilisée (i.e. (P3)) est généralement très forte. Le nombre moyen de sous-tours est très peu élevé dans la solution initiale. De plus, la valeur de Z_0 dans la plupart des cas est à l'intérieur de 1 % de Z^*. La différence $(Z^* - Z_0)$ diminue quand m

augmente ou quand m/n augmente. Ceci est dû au fait que quand m est large, plusieurs arcs doivent être incidents au dépôt. Les contraintes de degré deviennent plus fortes, de même, évidemment, que la relaxation;

(iv) la qualité de l'heuristique est aussi excellente. La valeur de Z_H se tient en plusieurs cas, à l'intérieur de quelques pourcentages de Z*. Encore une fois la différence (Z_H- Z*) diminue quand m augmente;

(v) le nombre de noeuds dans l'arbre d'énumération exploré (DESC) est assez petit. Cependant, le temps de résolution à chaque noeud est assez grand. À notre avis, l'algorithme RNET s'est avéré peu rapide. Un outil plus efficace pour résoudre des problèmes de flot à coût minimum aurait grandement amélioré la qualité de nos résultats.

Nous avons mis en annexe la liste FORTRAN qui correspond à l'algorithme décrit dans ce chapitre.

CHAPITRE 4

ALGORITHME EXACT BASÉ SUR LE PROBLÈME D'AFFECTATION

4.1 Modèle du PVC asymétrique

Le problème du voyageur de commerce (PVC) [2,20] est celui rencontré par un voyageur qui doit visiter n villes puis retourner à son point de départ, tout en minimisant sa distance totale parcourue. On supposera toujours que le PVC est défini par n villes dont n-1 doivent être visitées par le voyageur et une lui servant de dépôt, c'est-à-dire de point de départ et d'arrivée. Le dépôt sera toujours identifié à la ville 1. Les notations associées au PVC sont les mêmes que pour le m-PVC.

Le problème du PVC peut être formulé comme suit :

$$\text{Minimiser} \qquad Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} X_{ij} \tag{4.1}$$

assujettie à

$$\sum_{j=1}^{n} X_{ij} = 1 \quad , \quad i=1,\ldots,n \tag{4.2}$$

(P4)

$$\sum_{i=1}^{n} X_{ij} = 1 \quad , \quad j=1,\ldots,n \tag{4.3}$$

$$\sum_{\substack{i\in S \\ j\in S}} X_{ij} \leq |S| - 1 \quad \text{(pour tout } S \subseteq N, \; 2 \leq |S| \leq n-2) \tag{4.4}$$

$$X_{ij} = 0,1 \quad , \quad i,j=1,\ldots,n \tag{4.5}$$

où $(S,\bar{S})$ est une partition de l'ensemble des villes et $|S|$ représente la cardinalité de l'ensemble S. L'équation (4.1) évalue la distance totale parcourue par le voyageur. Les contraintes (4.2) et (4.3) constituent les contraintes de degré pour le PVC : elles contrôlent le nombre de mouvements d'entrées et de sorties dans chaque ville. Elles sont dénommées aussi les contraintes d'affectation.

Les contraintes (4.4) sont appelées contraintes d'élimination des sous-tours. Elles empêchent la formation de sous-tours illégaux qu'on pourrait obtenir en incluant dans (P4) seulement les contraintes d'affectation. La forme de ces contraintes est une génération directe des contraintes de Dantzig au cas du PVC, cas asymétrique.

Dans la section suivante, on verra la transformation du m-PVC asymétrique en un PVC asymétrique par de simples modifications de la matrice des distances. Dans la section 4.3.1, on démontre qu'en relaxant les contraintes d'élimination des sous-tours illégaux dans (P4) on obtient un problème d'affectation qui lui aussi est bien résolu en recherche opérationnelle. La section 4.3.2 donne une description de l'algorithme de "branch-and-bound" basée sur le problème d'affectation. La section 4.4 décrit des résultats des tests sur ordinateur concernant l'utilisation de cet algorithme lorsque m est fixé ou libre.

4.2 Transformation du m-PVC en un PVC

On peut toujours associer à un problème du type m-PVC un problème du type PVC par des simples modifications de la matrice des distances. Une des premières transformations de ce type peut être retrouvée dans la thèse de Hong [17]. On retrouvera aussi un résumé sur ce sujet dans [23]. La transformation que nous utiliserons ici est due à Lenstra et al. [21]. Elle est plus générale que celle de Hong [17] et nous en donnons une description détaillée dans ce qui suit.

Pour un m-PVC avec n+1 villes où la ville 1 sert de dépôt et où au plus $\bar{m}$ commis sont utilisés, la matrice des distances du PVC associée au m-PVC est décrite dans la figure 4.1.

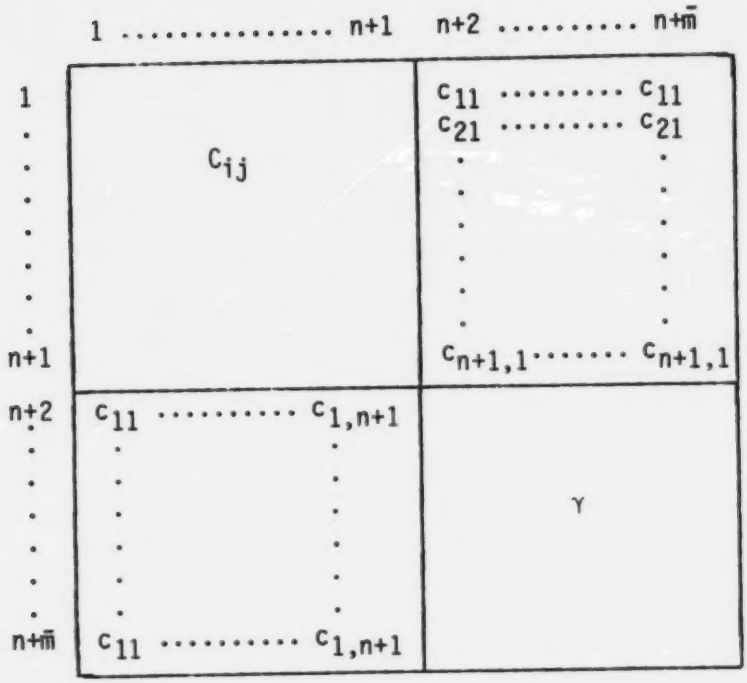


Figure 4.1 : Matrice des distances du PVC associée à un m-PVC

On rajoute donc au graphe du m-PVC $\bar{m}$-1 noeuds, appelés les villes artificielles (n+2, ..., n+$\bar{m}$). La distance entre une ville artificielle et une autre est γ tandis que la distance entre une des villes originale $i\in\{1,\ldots,n+1\}$ vers une des villes artificielles (ou l'inverse) est $c_{i,1}(c_{1,i})$, i.e. : un tel passage est considéré comme un retour (ou un départ du) au dépôt. Si on résout le PVC avec une telle matrice des distances et que le tour obtenu contienne $\bar{m}$-A liens entre les villes artificielles, dans la solution du m-PVC associée m sera égale à A. Cette transformation nous permet donc de résoudre par un PVC un problème du type m-PVC où m est libre. La constante γ peut être interprétée de la façon suivante : utiliser un véhicule ou un commis de plus, diminue de 1 le nombre de liens entre les villes aritificielles, d'où $-\gamma$ est le coût d'un véhicule. À partir d'une telle signification pour γ, on peut conclure que :

(i) si $\gamma=+\infty$, la solution du PVC nous fournit la distance minimum pour $\bar{m}$ commis utilisés (m est fixé à $\bar{m}$);

(ii) si $\gamma=-\infty$, la solution du PVC contient un nombre minimal de commis;

(iii) si $\gamma=0$, on a résolu par le PVC associé un m-PVC où m est libre.

On voit donc que la connaissance d'un algorithme pour le PVC nous permet de résoudre un m-PVC où m est libre ou fixé. Il faut noter cependant qu'une telle transformation possède le désavantage d'augmenter la taille du problème à résoudre et nécessite la connaissance d'une borne supérieure sur m (cas où m est libre).

Exemple : Un problème m-PVC avec cinq villes (donc n=4) et m=3 (m fixé) et matrice des distances :

	1	2	3	4	5
1	∞	2	3	1	0
2	1	∞	4	7	9
3	6	5	∞	2	8
4	1	2	3	∞	2
5	7	6	4	3	∞

On aura la matrice des distances du PVC associée comme suit :

	1	2	3	4	5	6	7
1	∞	2	3	1	0	∞	∞
2	1	∞	4	7	9	1	1
3	6	5	∞	2	8	6	6
4	1	2	3	∞	2	1	1
5	7	6	4	3	∞	7	7
6	∞	2	3	1	0	∞	∞
7	∞	2	3	1	0	∞	∞

La figure 4.2 illustre un 2-tour pour un 2-PVC avec n=4 et le tour qui lui est associé par la transformation de Lenstra et al. [21]. Les distances sur le 2-tour et le tour sont indiqués sur les arcs.

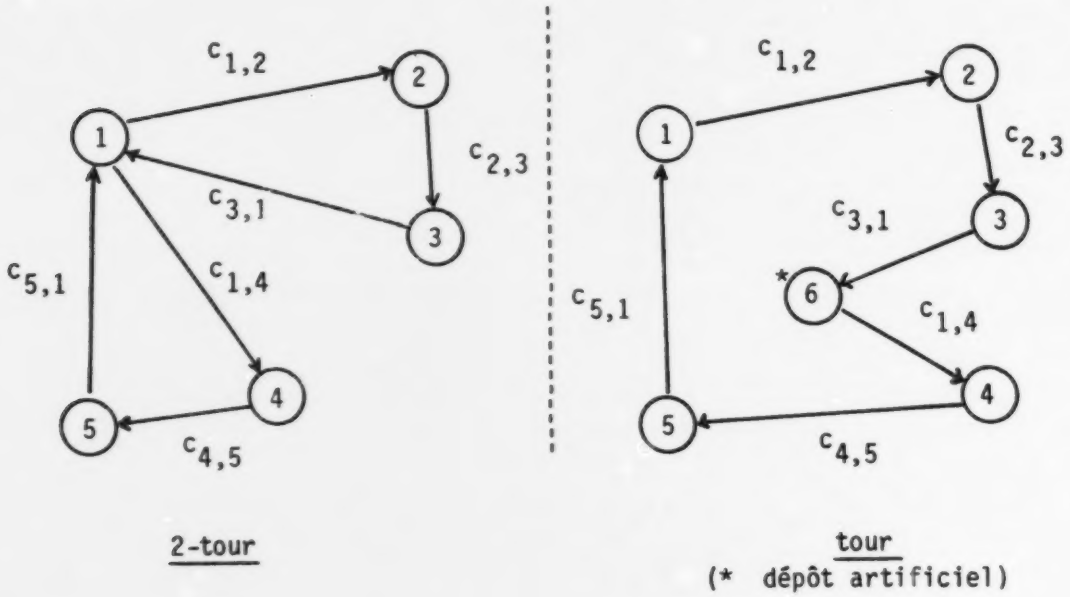


Figure 4.2 : Correspondance entre un 2-tour et un tour par la transformation de Lenstra et al.

L'intérêt d'effectuer une telle transformation est qu'elle nous permet de résoudre le m-PVC comme un PVC, toutefois ceci est effectué au détriment d'une augmentation du nombre de sommets dans le graphe associé au problème. De façon générale, cette thèse peut être résumée comme la comparaison entre une approche directe de résolution du m-PVC versus une approche indirecte par le biais d'une transformation. On discute en conclusion des avantages et des désavantages de ces deux approches.

4.3 Algorithme de "branch-and-bound" basé sur le problème d'affectation

4.3.1 Résolution de (P4) comme un problème d'affectation

Les contraintes d'élimination des sous-tours (4.4) rendent difficile la résolution le problème (P4) car elles sont beaucoup trop nombreuses pour en tenir compte explicitement. De plus, les contraintes

d'intégrité (4.5) des variables de (P4) ajoutent la touche finale à son extrême complexité. Toutefois, en ôtant les contraintes (4.4), on obtient un problème d'affectation (PA), problème qui se résout facilement.

Le domaine réalisable du PVC est inclus dans celui du (PA) associé, et dans le cas asymétrique, l'algorithme de Hitchcock nous permet de résoudre le (PA) aisément.

La solution optimale du (PA) est souvent très près de celle du PVC. Shapiro en 1966 [25] a effectué plusieurs tests qui corroborèrent cette affirmation.

4.3.2 Algorithme de "branch and bound"

Nous utilisons ici la même méthode de branchement de Carpaneto et Toth [6] que nous avons décrite dans le chapitre précédent. Sauf qu'ici, chaque noeud de l'arbre d'énumération correspond à un sous-problème d'affectation au lieu d'un sous-problème de flot. Le domaine réalisable du sous-problème d'affectation est un sous-ensemble du domaine réalisable du (PA) original associé au PVC. La racine de l'arbre correspond justement à ce dernier (PA).

Notons que tout d'abord, nous devons transformer le problème du m-PVC en un PVC comme nous avons décrit plus haut.

4.4 Résultats des tests

Dans le tableau 4.2, on retrouve les résultats des tests effectués avec l'algorithme de "branch and bound" basé sur le problème d'affectation lorsque m est fixé à 5, 10 pour n = 350, 400, 450, 500, 550, 600, 650, 700 villes. Ainsi que dans le tableau 4.3, on retrouve les résultats des tests lorsque m est libre avec $\bar{m} = 5$.

Tableau 4.2
Résultats des tests quand m est fixé

n	m	SUCCES	NB-SSTOURS	MOY-SSTOURS	NOEUDS	DESC	TEMPS
350	5	3/3	3	4.6	10.0	3.7	175.8
	10	3/3	3	3.6	9.0	4.7	51.9
400	5	3/3	3	3.0	20.3	3.0	171.8
	10	3/3	3	2.6	42.0	4.3	135.0
450	5	3/3	3	2.6	9.0	4.0	287.1
	10	3/3	3	2.0	14.0	1.7	252.8
500	5	3/3	3	3.0	12.7	2.7	123.5
	10	2/3	3	3.3	11.3	2.3	682.9
550	5	2/3	3	3.0	15.0	2.3	588.5
	10	3/3	3	2.6	13.7	2.6	231.8
600	5	3/3	3	2.6	18.7	1.0	317.7
	10	2/3	3	4.0	34.7	8.3	639.2
650	5	2/3	2	1.3	13.0	1.3	562.9
	10	2/3	3	5.0	14.7	2.7	819.2
700	5	2/3	3	3.0	44.3	4.0	491.6
	10	2/3	3	5.0	38.0	3.0	513.5
750	5	1/3	3	2.6	39.0	3.0	752.7
	10	0/3	3	2.3	35.0	2.0	1000.0

Tableau 4.3
Résultats des tests quand m est libre avec $\bar{m} = 5$

n	m	SUCCES	NB-SSTOURS	MOY-SSTOURS	NOEUDS	DESC	TEMPS
300	1 1 1	3/3	3	3.7	18.7	5.7	63.5
350	1 1 1	3/3	3	4.0	12.3	4.0	115.7
400	1 1 1	3/3	3	2.3	35.0	7.7	379.0
450	1 1 1	2/3	3	3.7	47.0	12.0	454.2
500	1 1 1	3/3	3	3.7	25.3	1.7	201.6
550	1 1 1	2/3	3	1.7	12.7	1.3	624.5
600	1 1 1	2/3	3	1.0	30.0	9.7	829.6
650	1 1 1	1/3	3	4.0	24.0	1.7	763.0
700	1 1 1	3/3	3	2.7	26.7	4.7	401.0
750	1 1 1	0/3	3	4.7	34.7	4.3	1000.0

Les c_{ij} sont générés aléatoirement dans l'intervalle [1,1 000] à l'aide d'une distribution uniforme. Ils sont tronqués à l'entier le plus près. Tous les problèmes sont résolus par l'ordinateur VAX de l'Université de Montréal en utilisant le compilateur fortran FTN. Le temps de résolution d'un problème est limité à 1 000 secondes.

La notation utilisée dans les tableaux 4.2 et 4.3 s'interprètent comme au chapitre 3 sauf dans le tableau 4.3 où m est le nombre de commis voyageurs trouvé par l'algorithme avec $\bar{m}$=5.

On peut tirer les conclusions suivantes à partir des tableaux 4.2 et 4.3 :

(i) l'algorithme s'est avéré très efficace. Il a pu résoudre des problèmes contenant 750 villes, taille qui, à notre connaissance, n'a jamais été atteinte jusqu'à maintenant.

(ii) lorsque m est fixé, on a rencontré le même phénomène qu'avec l'algorithme précédent : la difficulté du problème diminue quand m augmente.

(iii) la relaxation est généralement bonne et le nombre moyen de sous-tours est assez petit dans la solution initiale.

(iv) le nombre de noeuds dans l'arbre d'énumération exploré (DESC) est assez petit et le temps de résolution par noeud est très raisonnable.

Puisque l'algorithme décrit dans ce chapitre a été obtenu en modifiant de façon mineure un algorithme déjà construit pour le PVC, nous n'avons pas jugé utile d'en mettre la liste FORTRAN en annexe.

CONCLUSION

Dans ce mémoire nous avons décrit deux nouveaux algorithmes exacts pour le problème des m voyageurs de commerce (m-PVC). Le premier repose sur une nouvelle formulation mathématique du m-PVC et permet de résoudre de façon exacte ce dernier à l'aide d'un algorithme de "branch-and-bound" où, à chaque noeud de l'arbre d'énumération, un problème de flot à coût minimum est résolu. L'avantage marquant de cet algorithme sur ceux qui existent déjà dans la littérature vient du fait qu'il ne nécessite pas l'ajout de villes artificielles. Les résultats obtenus sur des problèmes engendrés aléatoirement avec cet algorithme furent excellents et des problèmes de 160 villes ont été résolus optimalement. Notons en outre que cet algorithme contient un nouvel heuristique pour le m-PVC très performant, basé sur la fusion efficiente de sous-tours par un calcul de pénalités.

Le deuxième algorithme optimal que nous avons construit constitue une adaptation pour le m-PVC de l'algorithme de Carpaneto et Toth [6] pour le PVC en utilisant la transformation de Lenstra et Rinnooy Kan [21]. En utilisant cet algorithme nous avons résolu des problèmes contenant jusqu'à 750 villes, taille à notre connaissance jamais atteinte dans la littérature. À notre avis ce deuxième algorithme s'est avéré plus performant que le premier uniquement à cause de la plus grande vitesse de résolution du problème d'affectation comparée à celle obtenue pour la résolution du problème de flot à coût minimum par RNET. Une extension possible du travail présenté dans ce mémoire serait d'améliorer l'algorithme résolvant la relaxation utilisée dans le premier algorithme. Nous obtiendrions alors un algorithme très performant pour le m-PVC.

BIBLIOGRAPHIE

[1] **Ali, A.I., Kennignton, J.L.,** "The M-travelling Salesman Problem: A Duality Based Branch-and-Bound Algorithm", Technical Report OR 80018, Department of Operations Research, Southern Methodist University, 1980.

[2] **Balas, E., Christofides, N.,** "A Restricted Lagrangean Approach to the Traveling Salesman Problem", Math. Programming 21, 19-46, 1981.

[3] **Bellmore, M., Hong, S.,** "Transformation of Multisalesman Problem to the Standard Travelling Salesman Problem", JACM 21, 500-504, 1974.

[4] **Bellmore, M., Malone, J.C.,** "Pathology of Traveling Salesman Subtour - Elimination Algorithms", Operations Research 19, 278-307, 1971.

[5] **Bodin, L.D., Golden, B.L., Assad, A.A., Ball, M.,** "Routing and Scheduling of Vehicles and Crews. The State of the Art", Computers and Operations Research 10, 62-212, 1983.

[6] **Carpaneto, G., Toth, P.,** "Some New Branching and Bounding Criteria for the Asymmetrical Travelling Salesman Problem", Management Science 26, 736-743, 1980.

[7] **Clarke, G., Wright, J.W.,** "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", Operations Research 12, 568-581, 1964.

[8] **Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.,** "Solution of a Large Scale Travelling Salesman Problem", Operations Research 2, 393-410, 1954.

[9] **Edmonds, J.,** "Maximum Matching and a Polyhedron with 0,1 Vertices", Journal of Research of the National Bureau of Standard, Section B69, 125-130, 1965.

[10] **Eilon, S., Watson-Gandy, C.D.T., Christofides, N.,** "Distribution Management: Mathematical Modelling and Practical Analysis", Griffin, 1971.

[11] **Garfinkel, R.S.,** "Branch-and-Bound Methods for Integer Programming", in: Combinatorial Optimization, N. Christofides et al. (Eds.), Wiley, 1979.

[12] **Gavish, B., Graves, S.C.,** "The Travelling Salesman Problem and Related Problems", Working paper No. 7905, Graduate School of Management, University of Rochester, 1979.

[13] **Gavish, G., Srikanth, K.,** "An Optimal Solution Method for the Multiple Travelling Salesman Problem", Working paper No. 8027, Graduate School of Management, University of Rochester, 1980.

[14] **Grigoriadis, M.D., Hsu, T.,** "RNET - The Rutgers Minimum Cost Network Flow Subroutines", Rutgers University, New Brunswick, N.J., 1979.

[15] **Held, M., Karp, R.M.**, "The Travelling Salesman Problem and Minimum Spanning Trees", Operations Research 18, 1138-1162, 1970.

[16] **Held, M., Karp, R.M.**, "The Travelling Salesman Problem and Minimum Spanning Trees: Part II", Mathematical Programming 1, 6-25, 1971.

[17] **Hong, S.**, "A Linear Programming Approach for the Travelling Salesman Problem", The Johns Hopkins University, Ph.D., 1972.

[18] **Hong, S., Padberg, M.W.**, "A Note on the Symmetric Multiple Traveling Salesman Problem with Fixed Charges", Operations Research 25, 871-873, 1977.

[19] **Laporte, G., Nobert, Y.**, "A Cutting Planes Algorithm for the m-Salesman Problem", Journal of the Operational Research Society 31, 1017-1023, 1980.

[20] **Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Schnoys, D.B.**, Eds., "The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization", Wiley, 1985.

[21] **Lenstra, J.K., Rinnooy Kan, A.H.G.**, "Some Simple Applications of the Travelling Salesman Problem", Operations Research Quarterly 26, 717-733, 1975.

[22] **Lin, S.**, "Computer Solutions of the Traveling Salesman Problem", Bell System Tech. J. 44, 2245-2269, 19 .

[23] **Nobert, Y.**, "Construction d'algorithmes optimaux pour des extensions au problème du voyageur de commerce", Université de Montréal, Ph.D., 1982.

[24] **Sandi, C.**, "Subgradient Optimization", Combinatorial Optimization, 73-91, Wiley, 1979.

[25] **Shapiro, D.**, "Algorithms for the Solution of the Optimal Cost and Bottleneck Travelling Salesman Problems", Washington University, Sc.D., 1966.

[26] **Smith, T.H.C.**, "A LIFO Implicit Enumeration Algorithm for the Asymmetric Travelling Salesman Problem using a One-Arborescence Relaxation", Mathematical Programming Study 12, 108-114, 1980.

[27] **Svestka, J.A., Huckfeldt, V.E.**, "Computational Experience with an m- Salesman Travelling Salesman Algorithm", Management Science 19, 790-799, 1973.

ANNEXE

LISTE FORTRAN POUR LE PREMIER ALGORITHME
(BASÉ SUR UN PROBLÈME DE FLOT À COÛT MINIMUM)

```
10=FLOT,CM300000,T100.
20=*CODE
30=LIB,L,TAPE5,N=DON20.
40=REWIND(TAPE5)
50=FTN(L=0,PL=100000)
60=LGO.
70=REWIND(TAPE9)
80=LIB,C,TAPE9,N=ESSAIMF.
90=EXIT(S)
100=REWIND(TAPE9)
110=LIB,L,TAPE9,N=ESSAIMF.
120=*WEOR
130=*FTN
140=*NUM
150=      PROGRAM CLCMPLT(INPUT,OUTPUT,TAPE1=OUTPUT,TAPE5,TAPE4=INPUT,TAPE9)
160=C$    DEBUG
170=C$    ARRAYS
180=      INTEGER FROM,TO,F,D,P,R
190=      INTEGER COUT,CAPAC
200=      INTEGER GENARC,DIST,C
210=      INTEGER   BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
220=      INTEGER  SOURCE, DESTIN,PLM, NARC
230=      INTEGER   TR(42,180), STR(12,180)
240=      INTEGER NUMARC(1000),VALBASE
250=      INTEGER IA, QUEUE(20),TABZ(1000),TBRANC(10), TNOARC(180),
260=     &  EA
270=      INTEGER DIST1J,DISTJ1,DISTAMP(180)
280=      DIMENSION IA(1000), EA(1000)
290=      INTEGER CNOARC(180), TABCHOI(10)
300=      DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
310=      DIMENSION PCAND(1)
320=      DIMENSION RTN(20),LPR(12),RTT(2)
330=      COMMON /LIMITE/ MXM,MXN
340=      COMMON // GENARC(33 000)
350=      COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
360=      COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
370=     *         NCAND
380=      DIMENSION IREVFL(4)
390=      COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
400=     *         ISD8,TEO
410=      COMMON DIST1J(180),DISTJ1(180),NBVILLE
420=      C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
430=      FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
440=      H(I)=SHIFT(GENARC(I),-50)
450=      TO(I)=SHIFT(GENARC(I),-30).AND.1777B
460=*
470=*
480=*  NBVILLE  :  NOMBRE DE VILLES INITIALE
490=*  NVOY     :  NOMBRE DE VOYAGEURS
500=*  M        :  NOMBRE DE NOEUDS
510=*  N        :  NOMBRE D'ARCS
520=*   C      : COUT D'UNITE DONNE (GIVEN UNIT-COST)
530=*
540=      NBVILLE = 20
550=*      NCOMMIS(1) = 2
560=*      NCOMMIS(2) = 5
570=*      NCOMMIS(3) = 10
580=*      NCOMMIS(4) = 20
590=*      NCOMMIS(5) = 40
600=      NBV = NBVILLE - 1
610=      BT = 30
620=      PT = 0
630=*     NOMBRE DE NOEUDS)
640=      M = NBV + NBV + 2
```

```
650=*
660=*       NOMBRE D'ARCS = ARCS ENTRE NBV VILLES + SOURCE --> NBV +
670=*       + NBV --> DESTIN + DESTIN --> SOURCE
680=*
690=*        NNBA = NBV * (NBV - 1) + NBV + NBV + 1
700=        IRT = 1
710=        FRQ = 2.1
720=        MXIT = 20000
730=        PO=1.0
740=        PBAR = 1.5
750=        NCAND = 0
760=*       MAX DE NOEUD
770=        MXM = 2000
780=*            MAX D'ARCS
790=        MXN = 30000
800=*       LIRE LES DONNEES
810=        DO 4 PLM=1,3
820=        NVOY = 2
830=        PRINT *," LE PROBLEME NUMERO :  ",PLM," NBR.  DE VOY.  :",NVOY
840=       +   ," NBR DE VILLES    ",NBVILLE
850=        LONGINI = 0
860=        NARC = 0
870=        N = 0
880=        NN = NBVILLE +2
890=        NN2 = NBVILLE + NBVILLE
900=*        DO 5 I = 1,NBVILLE
910=        READ(5,12) (DIST1J(J),J=1,NBVILLE)
920=*        PRINT *," DIST1J ",(DIST1J(J),J=1,NBVILLE)
930=        DIST1J(1) = 999999
940=        DISTJ1(1) = 999999
950=*        PRINT *, (DIST(I,J),J=1,NBVILLE)
960=*        DIST(I,I) = 999999
970=*     5 CONTINUE
980=     12 FORMAT(" ",25(I3))
990=*        DO 450 I9=2,NBVILLE
1000=*        LONGINI = LONGINI + DIST(1,I9) + DIST(I9,1)
1010=**        PRINT *," LONG + DIST + DIST ",LONGINI
1020=**       &   ," DIST(1,I) ",DIST(1,I9)," DIST(I,1) ",DIST(I9,1)
1030=*    450 CONTINUE
1040=*        PRINT *," LONGINI AU DEBUT ",LONGINI
1050=*        DO 7 NM=1,1
1060=*        NVOY = NCOMMIS(NM)
1070=*
1080=*             REMPLIR ARCS ENTRE NOEUD1 (A GAUCHE), NOEUD2 (A DROITE)
1090=*
1100=        DO 10 NOEUD1 = 2,NBVILLE
1110=        READ(5,12) (DISTAMP(J),J=1,NBVILLE)
1120=*        PRINT *," DISTAMP ",(DISTAMP(J),J=1,NBVILLE)
1130=        DISTJ1(NOEUD1) = DISTAMP(1)
1140=        LONGINI = LONGINI + DIST1J(NOEUD1) + DISTAMP(1)
1150=        DO 100 NOEUD2=NN,NN2
1160=        IF (NOEUD2 .EQ. NOEUD1 + NBVILLE) GO TO 100
1170=        NARC = NARC + 1
1180=*        FROM(NARC) = NOEUD1
1190=*        TO(NARC) = NOEUD2
1200=*        COUT = DIST(NOEUD1,NOEUD2 - NBVILLE) - DIST(NOEUD1,1) -
1210=*       +    DIST(1,NOEUD2 - NBVILLE)
1220=*        C(NARC) = SIJ
1230=*           CAPACITE - BORNE INF.
1240=*        H(NARC) = 1 - 0
1250=        COUT = DISTAMP(NOEUD2-NBVILLE) - DISTAMP(1) -
1260=       & DIST1J(NOEUD2 - NBVILLE)
1270=*        PRINT *," COUT ",COUT
1280=        CAPAC = 1
1290=        CALL AJARC(NOEUD1,NOEUD2,COUT,CAPAC)
1300=*        TABSIJ(NARC,1) = (NOEUD1 * 1000) + NOEUD2
1310=*        TABSIJ(NARC,2) = COUT
1320=*        PRINT *," TABSIJ ",TABSIJ(NARC,1),TABSIJ(NARC,2)
```

```
1340=    10 CONTINUE
1350=       PRINT *," LONGINI AU DEBUT ",LONGINI
1360=*       DO 15 LIG=1,NBVILLE
1370=*       DO 20 NCOL=1,NBVILLE
1380=*       PRINT *," DIST ",DIST(LIG,NCOL)
1390=*    20 CONTINUE
1400=*    15 CONTINUE
1410=*
1420=*-----------------------------------------------------------------
1430=*       CALL HEAP(TABSIJ,NARC)
1440=*       PRINT *," SAU KHI TRIE "
1450=*       DO 105 I4=1,NARC
1460=*       PRINT *,TABSIJ(I4,1),TABSIJ(I4,2)
1470=*   105 CONTINUE
1480=*       N = (NBVILLE * NBVILLE) * .2
1490=*       CAPAC = 1
1500=*       DO 110 I4=1,N
1510=*       NOEUD1 = TABSIJ(I4,1) / 1000
1520=*       NOEUD2 = TABSIJ(I4,1) - (NOEUD1 * 1000)
1530=*       PRINT *," NOEUD1,NOEUD2 ",NOEUD1,NOEUD2
1540=*       CALL AJARC(NOEUD1,NOEUD2,TABSIJ(I4,2),CAPAC)
1550=*   110 CONTINUE
1560=*-----------------------------------------------------------------
1570=*
1580=       DO 200 NOEUD1 = 2,NBVILLE
1590=       X(NOEUD1) = 0
1600=       X(NOEUD1+NBVILLE) = 0
1610=   200 CONTINUE
1620=*
1630=*            COMME BORNE INF = 0 ALORS PAS DE MODIFICATION COMME
1640=* SUGERE DANS RNET
1650=*
1660=*
1670=*            REMPLIR NARCS ENTRE SOURCE ET NOEUD DE SORTIE
1680=*
1690=       SOURCE = 1
1700=       X(SOURCE) = 0 + NBVILLE - NVOY - 1
1710=       DO 300 NOEUD1 = 2,NBVILLE
1720=       NARC = NARC + 1
1730=*       FROM(NARC) = SOURCE
1740=*       TO(NARC) = NOEUD1
1750=*       C(NARC) = 0
1760=*       H(NARC) = 1 - 0
1770=       COUT = 0
1780=       CAPAC = 1
1790=       CALL AJARC(SOURCE,NOEUD1,COUT,CAPAC)
1800=   300 CONTINUE
1810=*
1820=*
1830=*
1840=       DESTIN = NBVILLE + 1
1850=       X(DESTIN) = 0 - NBVILLE + NVOY + 1
1860=*
1870=*         REMPLIR NARCS ENTRE NOEUDS DE SORTIE ET DESTIN
1880=*
1890=       DO 400 NOEUD2=NN,NN2
1900=       NARC = NARC + 1
1910=*       FROM(NARC) = NOEUD2
1920=*       TO(NARC) = DESTIN
1930=*       C(NARC) = 0
1940=*       H(NARC) = 1 - 0
1950=       COUT = 0
1960=       CAPAC = 1
1970=       CALL AJARC(NOEUD2,DESTIN,COUT,CAPAC)
1980=   400 CONTINUE
1990=*
```

```
2000=*
2010=       NARC = NARC + 1
2020=*       FROM(NARC) = DESTIN
2030=*       TO(NARC) = SOURCE
2040=*       C(NARC) = 0
2050=*       H(NARC) = 0
2060=       COUT = 0
2070=       CAPAC = 0
2080=       CALL AJARC(DESTIN,SOURCE,COUT,CAPAC)
2090=*       PRINT *," CALL RNWIPA : IMPRIME LA VALEUR INITIALE DE PARA."
2100=*       CALL RNWIPA(1)
2110=*       PRINT *," CALL RNET "
2120=*       PRINT *," NBR D'ARCS TOTAL ", NARC
2130=       CALL SECOND(T1)
2140=       CALL RNET
2150=       CALL SECOND(T2)
2160=       PRINT *," RTN(1) ",RTN(1)," RTN(2) ",RTN(2)," RTN(4) ",
2170=      &  RTN(4),"  RTN(5) ",RTN(5)," RTN(6) ",RTN(6)
2180=       WRITE(9,*) "RTN(1)  ",RTN(1)," RTN(2)  ",RTN(2)
2190=*       PRINT *," CALL RNWFPA "
2200=*       CALL RNWFPA(1)
2210=*       PRINT *," CALL RNSOL :ARCS DE BASE "
2220=*       CALL RNSOL(1,1,1,1,1,N,1,M,NHITS)
2230=*
2240=*-------------------------------------------------------------------
2250=**       PRINT *," CALL RNSOL : ARCS HORS BASE "
2260=*       CALL RNSOL(1,3,1,1,1,N,1,M,NHITS)
2270=*       NCFIXE = 0
2280=*       DO 500 L4=1,N
2290=*       FLOT = -H(L4)
2300=*       IF (FLOT .LT. 0) GO TO 490
2310=*       PRINT *," FROM ",FROM(L4)," TO ",TO(L4)
2320=*       GO TO 500
2330=*  490   FLOT = 0
2340=*       NOEUD = FROM(L4)
2350=*       IF (ARC(NOEUD) .EQ. L4) GO TO 500
2360=*       NOEUD = TO(L4)
2370=*       IF (ARC(NOEUD) .EQ. L4) GO TO 500
2380=*       PRINT *," FROM(I) ",FROM(L4)," TO(I) ",TO(L4)
2390=*  500 CONTINUE
2400=*-------------------------------------------------------------------
2410=*
2420=*       LTEST = 0
2430=*       DO 460 I9=2,NBVILLE
2440=*       LTEST = LTEST + DIST(1,I9) + DIST(I9,1)
2450=*       PRINT *," LONG + DIST + DIST ",LTEST
2460=*      &  ," DIST(1,I) ",DIST(1,I9)," DIST(I,1) ",DIST(I9,1)
2470=*  460 CONTINUE
2480=*       PRINT *," LTEST AU DEBUT ",LTEST
2490=      LONG = LONGINI + RTN(2)
2491=      TTTRNET = T2 - T1
2500=      PRINT *," LONGEUR DU TOUR : ",LONG," TEMPS DE RESOL. ",TTTRNET
2510=*       WRITE(9,*)" LONGEUR DU TOUR : ",LONG," TEMPS DE RESOL. ",TTTRNE
2520=*
2530=       CALL CHECKST(TR,STR,NBSTR,NBM)
2540=       IF (NBSTR .EQ. 0) GO TO 7
2541=       CALL RNSOL(1,3,1,1,1,N,1,M,NHITS)
2550=       CALL SECOND(TB1)
2560=       LONGSUP = LONG
2570=       CALL HRISTIG(NBSTR,NBM,TR,STR,LONGSUP)
2580=       NCFIXE = 0
2590=*          ----------------------------------------
2600=*          ELIMINER A L'AIDE DES COUTS RELATIFS
2610=*          ----------------------------------------
2620=*
2630=      LDIF = LONGSUP - LONG
2640=      CALL FIXER(LDIF,NCFIXE)
2650=      CALL RNREV
```

```
2660=       NCFIXE = NCFIXE * 999
2670=       CALL RNREST
2680=       PRINT *," RTN(1) ",RTN(1)," RTN(2) ",RTN(2)," RTN(4) ",
2690=      &  RTN(4),"  RTN(5) ",RTN(5)," RTN(6) ",RTN(6)
2700=*      PRINT *," RTN + NCFIXE ",RTN(2) + NCFIXE
2710=*  ------------------------------------------------
2720=*
2730=*        ----------------------------------------
2740=*        PARTIE  B. AND B.
2750=*        ----------------------------------------
2760=*
2770=       IIA = 0
2780=       IEA = 0
2790=       NZETOIL = LONGSUP
2800=       NIVEAU = 1
2810=       NPTG = 0
2820=       VALBASE = 0
2830=       NPROB = 0
2840=       NPTIA2 = 0
2850=       NPTZ1 = 1
2860=       NBNEXAM = 0
2870=       NBNDSG = 0
2871=       NZCHOI = LONG
2873=       NFOI = 1
2880=*
2890=*
2900=*
2910=   600 CONTINUE
2920=*
2930=*        CHERCHER LA COMPOSANTE AYANT LE PLUS PETIT NOMBRE D'ARCS
2940=*        NON FIXES A 1
2950=*
2960=       NUMST = 1
2970=       NOMBARC = STR(1,1)
2980=       IF (NBSTR .EQ. 1) GO TO 702
2990=       CALL CMPPPNA(STR,NBSTR,NUMST,NOMBARC)
3000=       PRINT *," COMPOSANTE ",NUMST," NB ARCS ",NOMBARC
3010=*
3020=*        RECHERCHER LES NUMEROS DES ARCS DANS SS-TR  "NUMST"
3030=*
3040=   702 CALL RECNOA(TNOARC,STR,NUMST,CNOARC)
3050=*       PRINT *," TNOARC ",(TNOARC(I4),I4=1,NOMBARC)
3060=*      &    ," CNOARC ",(CNOARC(I4),I4=1,NOMBARC)
3070=       NPTIA1 = NPTIA2 + 1
3080=*
3090=*        CREATION  IA  ET EA
3100=*
3110=       CALL CRIAEA(IIA,IEA,IA,EA,NOMBARC,TNOARC,CNOARC)
3120=*       PRINT *," IIA ",IIA," IEA ",IEA
3130=*      &  ," CNOARC ",(CNOARC(I5),I5=1,NOMBARC)
3140=*       PRINT *," IA ",(IA(I4),I4=1,IIA)," EA ",(EA(I4),I4=1,IEA)
3150=       NPTIA2 = IIA
3160=       PRINT *," NPTIA1 ",NPTIA1," NPTIA2 ",NPTIA2
3170=       TBRANC(NIVEAU) = NOMBARC
3180=*       PRINT *," TBRANC ",(TBRANC(I4),I4=1,NIVEAU)
3190=       NDESC = 0
3200=       CALL ETAPE4(NIVEAU,NOMBARC,LONGINI,NZETOIL,EA,IA,IEA
3210=      &  ,TABZ,NPROB,NUMARC,VALBASE,NPTIA1,NDESC,NBNDSG,NCFIXE,NZCHOI,
3211=      &  TTTRNET,NFOI)
3220=*       PRINT *," NPROB ",NPROB," NPTZ1 ",NPTZ1," NPTG ",NPTG
3230=*      &    ," NDESC ",NDESC,"QUEUE "
3240=       IF (NDESC .EQ. 0) GO TO 1000
3250=       NIVEAU = NIVEAU + 1
3260=       CALL RNREV
3270=       CALL RPROBP(NPTIA1,NPTIA2,IA)
3280=*       PRINT *,"   NPROB  ", NPROB
3290=       IF (NPROB .EQ. 0) GO TO 8
3300=       NZCHOI = 9999
```

```
4580=       SUBROUTINE TRIER(TABMIN, INDI, INDJ, L, NT, IMIN, JMIN, STMIN)
4590=       INTEGER NT(10), TABMIN(10), INDI(10), INDJ(10), L
4600=       INTEGER IMIN(10), JMIN(10), STMIN(10)
4610=       DO 100 IT=2,L
4620=       IX=TABMIN(IT)
4630=       IP=INDI(IT)
4640=       IQ=INDJ(IT)
4650=       INST = NT(IT)
4660=       II = IMIN(IT)
4670=       IJ = JMIN(IT)
4680=       IS = STMIN(IT)
4690=       J = IT - 1
4700=  200   IF ((J .LE. 0) .OR. (TABMIN(J) .LE. IX)) GO TO 390
4710=       TABMIN(J+1) = TABMIN(J)
4720=       INDI(J+1) = INDI(J)
4730=       INDJ(J+1) = INDJ(J)
4740=       NT(J+1) = NT(J)
4750=       IMIN(J+1) = IMIN(J)
4760=       JMIN(J+1) = JMIN(J)
4770=       STMIN(J+1) = STMIN(J)
4780=       J = J - 1
4790=       GO TO 200
4800=  390   TABMIN(J+1) = IX
4810=       INDI(J+1) = IP
4820=       INDJ(J+1) = IQ
4830=       NT(J+1) = INST
4840=       IMIN(J+1) = II
4850=       JMIN(J+1) = IJ
4860=       STMIN(J+1) = IS
4870=   100 CONTINUE
4880=       RETURN
4890=       END
4900=*-----------------------------------------------------------------
4910=       SUBROUTINE LIER1(SSTR, TOUR, J1, NUMT, IK1, KS)
4920=       INTEGER SSTR(12,180), TOUR(42,180), J1, NUMT
4930=*       PRINT *," NOI O DAU ", IK1
4940=*       WRITE(9,*)" NOI O DAU ", IK1
4950=       IK2 = 3
4960=       DO 200 J=J1, IK1
4970=       TOUR(NUMT, J) = SSTR(KS, IK2)
4980=       IK2 =IK2 + 1
4990=  200   CONTINUE
5000=       RETURN
5010=       END
5020=*-----------------------------------------------------------------
5030=       SUBROUTINE LIER2(SSTR, TOUR, J1, NUMT, IK1, IK2, KS)
5040=       INTEGER SSTR(12,180), TOUR(42,180), J1, NUMT
5050=*       PRINT *," NOI O CUOI "," IK1 ", IK1
5060=*       WRITE(9,*)" NOI O CUOI "," IK1 ", IK1
5070=       DO 100 J=J1, IK1
5080=       TOUR(NUMT, J) = SSTR(KS, IK2)
5090=       IK2 = IK2 + 1
5100=  100   CONTINUE
5110=       RETURN
5120=       END
5130=       SUBROUTINE LIER3(SSTR, TOUR, J1, IT1, M, KS, IK)
5140=       INTEGER SSTR(12,180), TOUR(42,180), J1, IT1, M
5150=*       PRINT *," J1, IT1, M, KS ", J1, IT1, M, KS
5160=       JJ = J1
5170=  200   JJ = JJ + 1
5180=       IF (SSTR(KS, JJ) .NE. IK) GO TO 200
5190=       JJ = JJ + 1
5200=       IN2 = IT1
5210=  270   TOUR(M, IN2) = SSTR(KS, JJ)
5220=       IN2 = IN2 + 1
5230=       JJ = JJ + 1
5240=       IF (JJ .LE. SSTR(KS, 1)+2) GO TO 270
5250=       JJ = J1 + 1
5260=  230   IF (SSTR(KS, JJ) .EQ. TOUR(M, IT1)) GO TO 250
5270=       TOUR(M, IN2) = SSTR(KS, JJ)
5280=       IN2 = IN2 + 1
5290=       JJ = JJ + 1
5300=       GO TO 230
5310=  250   RETURN
5320=       END
5330=*-----------------------------------------------------------------
```

```
      IF (TB2 - TB1 .GT. 300.) GO TO 9
      NZCHOI = 9999
      DO 1100 I2=NPTZ1,NPROB
      IF (NZETOIL .LE. TABZ(I2)) TABZ(I2) = 9999
      IF (NZCHOI .LE. TABZ(I2)) GO TO 1100
      NZCHOI = TABZ(I2)
      NPCHOI = I2
 1100 CONTINUE
      IF (NZCHOI .EQ. 9999) GO TO 2000
      NIVEAU = NIVEAU + 1
*      PRINT *," PROBLEME A BRANCHE, RTN(1), RNT(2) ",RTN(1),RTN(2)
      GO TO 755
 2000  IF (NIVEAU .EQ. 1) GO TO 8
      CALL RNREV
      GO TO 1012
    9  CONTINUE
      PRINT *," TEMPS DEPASSE 300 SECONDES "
    8 CONTINUE
      CALL SECOND(TB2)
      PRINT *," SOLUTION TROUVEE ",NZETOIL," TEMPS DE B.B.  ",
     &    TB2 - TB1
      WRITE(9,*)" SOLUTION TROUVEE ",NZETOIL," TEMPS DE B.B.  ",
     &    TB2 - TB1
      PRINT *," NBR. DE NOEUDS DANS LA QUEUE AU TOTAL ",NBNDSQ
      PRINT *," NBR. DE NOEUDS ONT ETE EXAMINES ",NBNEXAM
      PRINT *," TEMPS TT. RNETS ",TTTRNET," TEMPS MOYEN ",
     &    TTTRNET / NFOI
    7 CONTINUE
    4 CONTINUE
      STOP
      END
*-------------------------------------------------------------
      SUBROUTINE LIAISON(TPQ,TR,SSTR,KS,QM,IK)
      INTEGER  TR(42,180), SSTR(12,180), QM
      INTEGER  TPQ
*      PRINT *," KS, IK",KS, IK," NOTR(1)",TPQ
      I1 = TR(TPQ,1) + SSTR(KS,1) + 2
      I2 = TR(TPQ,1) + 2
      I3 = I1 - I2
*      PRINT *," I1 , I2 , I3 ",I1,I2,I3
 3002 TR(TPQ,I1) = TR(TPQ,I2)
      IF (TR(TPQ,I2) .EQ. QM) GO TO 3100
      I1 = I1 - 1
      I2 = I2 - 1
      GO TO 3002
 3100  IF (SSTR(KS,2) .EQ. IK) GO TO 3200
      IF (SSTR(KS,SSTR(KS,1) + 1) .EQ. IK) GO TO 3300
      CALL LIER3(SSTR,TR,2,I1-I3,TPQ,KS,IK)
      GO TO 3500
 3200 IK1 = I1 - I3 + SSTR(KS,1) - 1
       CALL LIER1(SSTR,TR,I1-I3,TPQ,IK1,KS)
      GO TO 3500
 3300 IK1 = I1 - I3 + SSTR(KS,1) - 1
      IK2 = 2
       CALL LIER2(SSTR,TR,I1-I3,TPQ,IK1,IK2,KS)
 3500  TR(TPQ,1) = TR(TPQ,1) + SSTR(KS,1)
      IFIN = TR(TPQ,1) + 2
*       PRINT *," TR LIE T ",(TR(TPQ,I4),I4=1,IFIN)
*       WRITE(9,*)" TR LIE T ",(TR(TPQ,I4),I4=1,IFIN)
      RETURN
      END
*-------------------------------------------------------------
```

```
3310=*
3320=*      CHOISIR LE NOEUD AVEC LA PLUS PETITE VALEUR Z
3330=*
3340=       DO 750 I2=NPTZ1,NPROB
3350=       IF (NZETOIL .LE. TABZ(I2)) TABZ(I2) = 9999
3360=       IF (NZCHOI .LE. TABZ(I2)) GO TO 750
3370=       NZCHOI = TABZ(I2)
3380=       NPCHOI = I2
3390=  750   CONTINUE
3400=       IF (NZCHOI .EQ. 9999) GO TO 8
3410=  755  TABZ(NPCHOI) = 9999
3420=       NBNEXAM = NBNEXAM + 1
3430=       VALBASE = (NUMARC(NPCHOI) - 1) * 900 + VALBASE
3440=       PRINT *," NPCHOI   ",NPCHOI," VALBASE ",VALBASE," NZCHOI "
3441=      &  ,NZCHOI
3450=       TABCHOI(NIVEAU - 1) = NUMARC(NPCHOI)
3460=*
3470=*      FORMER LE PROBLEME PERE
3480=*
3490=       IF (NUMARC(NPCHOI) .EQ. 1) GO TO 810
3500=       IMOINS1 = NUMARC(NPCHOI) + NPTIA1 - 2
3510=       DO 800 I2=NPTIA1,IMOINS1
3520=       CALL RNRVCJ(IA(I2),-900)
3530=*       PRINT *," C DE 800 ",C(IA(I2))
3540=  800  CONTINUE
3550=  810   CALL RNRVCJ(EA(NPCHOI),999)
3560=       NPTQ = NPTQ + 1
3570=       QUEUE(NPTQ) = EA(NPCHOI)
3580=*       PRINT *," C DE EA ",C(EA(NPCHOI))
3590=*      &   ,  " QUEUE ",(QUEUE(I7), I7=1,NPTQ)
3600=       CALL SECOND(TN1)
3610=       CALL RNREST
3620=       CALL SECOND(TN2)
3630=       PRINT *," RTN1 ",RTN(1)," RTN2 ",RTN(2),
3640=      &   " TEMPS DE RNREST ",TN2 - TN1
3650=*       NPROB = NPROB - 1
3660=       CALL CHECKST(TR,STR,NBSTR,NBM)
3670=       LONGSUP = LONGINI + RTN(2) + (900 * NPCHOI)
3680=       NPTZ1 = NPROB + 1
3690=       GO TO 600
3700= 1000   IF (NIVEAU .EQ. 1) GO TO 8
3710=       CALL RNREV
3720=*       PRINT *," NPTIA1,   NPTIA2 ",NPTIA1,NPTIA2
3730=       CALL RPROBP(NPTIA1,NPTIA2,IA)
3740=*       CALL RNREST
3750=*       PRINT *," RTN(1), RTN(2), NOEUD 207  ",RTN(1),RTN(2)
3760=*       CALL RNREV
3770= 1012  NPTIA2 = NPTIA1 - 1
3780=       NPTIA1 = NPTIA1 - TBRANC(NIVEAU- 1) + 1
3790=       IIA = IIA - TBRANC(NIVEAU) + 1
3800=       IEA = IEA - TBRANC(NIVEAU)
3810=       NPROB = NPROB - TBRANC(NIVEAU)
3820=       NPTZ1 = NPTZ1 - TBRANC(NIVEAU -1)
3830=       VALBASE = VALBASE - ((TABCHOI(NIVEAU - 1) - 1) * 900)
3840=*       PRINT *," QUEUE(NPTQ) ",QUEUE(NPTQ)
3850=*      & ," NPTIA1, NPTIA2 , IIA, IEA, NPROB, NPTZ1, "
3860=*      &  , NPTIA1, NPTIA2, IIA,  IEA, NPROB, NPTZ1
3870=*      &   ," VALBASE ",VALBASE
3880=       CALL RNRVCJ(QUEUE(NPTQ),-999)
3890=       NPTQ = NPTQ - 1
3900=       IF (TABCHOI(NIVEAU - 1) .EQ. 1) GO TO 1015
3910=       IMOINS1 = TABCHOI(NIVEAU - 1) + NPTIA1 - 2
3920=       CALL RPROBP(NPTIA1,IMOINS1,IA)
3930= 1015   CALL RNREST
3940=*       PRINT *," RTN(1), RTN(2), NOEUD 201 ",RTN(1),RTN(2)
3950=       NIVEAU = NIVEAU - 1
3960= 1020  IF (NIVEAU .EQ. 0) GO TO 8
3970=       CALL SECOND(TB2)
```

```
5340=       SUBROUTINE RCP2KT(KTL,KL,TOUR,SSTR,MIN2,PMIN,GMIN,IMIN1,
5350=      &  JMIN1, NTR,MI)
5360=*
5370=*     RECALCULER P2K D'UN SS-TR (INDICE KL) AVEC LES TOURS
5380=*
5390=       INTEGER TOUR(42,180), SSTR(12,180), MIN2(10), PMIN(10), GMIN(10)
5400=       INTEGER IMIN1(10), JMIN1(10), NTR(10)
5410=       INTEGER FROM,TO,F,D,P,R
5420=       INTEGER GENARC,DIST,DIST1J,DISTJ1,C
5430=       INTEGER   BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
5440=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
5450=       DIMENSION PCAND(1)
5460=       DIMENSION RTN(20),LPR(12),RTT(2)
5470=       COMMON /LIMITE/ MXM,MXN
5480=       COMMON // GENARC(33 000)
5490=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
5500=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
5510=      *          NCAND
5520=       DIMENSION IREVFL(4)
5530=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
5540=      *          ISDB,TEO
5550=       COMMON DIST1J(180),DISTJ1(180),NBVILLE
5560=*       PRINT *," KL ",KL," MI ",MI
5570=       NAS = SSTR(KTL,1) + 1
5580=       MIN2(KL) = 9999
5590=       DO 100 I2=2,NAS
5600=       IK = SSTR(KTL,I2)
5610=       JK = SSTR(KTL,I2 + 1)
5620=       DO 200 IT3=1,MI
5630=       NAT = TOUR(IT3,1)
5640=       DO 300 IA=1,NAT
5650=       PENAL2 = DIST(TOUR(IT3,IA+1),JK)
5660=      &   + DIST(IK,TOUR(IT3,IA+2))
5670=      &   - DIST(IK,JK)
5680=      &   - DIST(TOUR(IT3,IA+1),TOUR(IT3,IA+2))
5690=       IF (MIN2(KL) .LE. PENAL2) GO TO 300
5700=       MIN2(KL) = PENAL2
5710=       PMIN(KL) = TOUR(IT3,IA+1)
5720=       GMIN(KL) = TOUR(IT3,IA+2)
5730=       IMIN1(KL) = IK
5740=       JMIN1(KL) = JK
5750=       NTR(KL) = IT3
5760=*       PRINT *," PENAL2 ",PENAL2," PMIN ",PMIN(KL)," GMIN ",GMIN(KL),
5770=*      &    " IK ",IK," JK ",JK," IT3 ",IT3
5780=   300 CONTINUE
5790=   200 CONTINUE
5800=   100 CONTINUE
5810=       RETURN
5820=       END
```

```
5830=*-------------------------------------------------------------------
5840=       SUBROUTINE RCP2KS(L1,L9,SSTR,KTR,MINS,PSMIN,QSMIN,IMIN2,JMIN2,
5850=      &  NBST)
5860=*
5870=*        RECALCULER P2K D'UN SS-TR L9 AVEC LES AUTRES SS-TR
5880=*
5890=       INTEGER SSTR(12,180), MINS(10), PSMIN(10), QSMIN(10)
5900=       INTEGER KTR(10), IMIN2(10), JMIN2(10)
5910=       INTEGER FROM,TO,F,D,P,R
5920=       INTEGER GENARC,DIST,DIST1J,DISTJ1,C
5930=       INTEGER   BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
5940=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
5950=       DIMENSION PCAND(1)
5960=       DIMENSION RTN(20),LPR(12),RTT(2)
5970=       COMMON /LIMITE/ MXM,MXN
5980=       COMMON // GENARC(33 000)
5990=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
6000=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
6010=      *          NCAND
6020=       DIMENSION IREVFL(4)
6030=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
6040=      *          ISDB,TEO
6050=       COMMON DIST1J(180),DISTJ1(180),NBVILLE
6060=*       PRINT *," L1",L1," L9 ",L9
6070=       NASSTR = SSTR(L1,1) + 1
6080=       DO 1870 L2=2,NASSTR
6090=       IK = SSTR(L1,L2)
6100=       JK = SSTR(L1,L2+1)
6110=       DO 1889 L4=1,NBST
6120=*       PRINT *," STR(L4,1) ",SSTR(L4,1)," L4 ",L4
6130=       IF (SSTR(L4,1) .EQ. 0) GO TO 1889
6140=       IF (L4 .EQ. L1) GO TO 1889
6150=       NAAST = SSTR(L4,1)
6160=*       PRINT *," L4 ",L4,"L1 ",L1," IK",IK," JK ",JK," NA ",NAAST
6170=       DO 1895 IA=1,NAAST
6180=       PENAL2 = DIST(SSTR(L4,IA+1),JK)
6190=      &   + DIST(IK,SSTR(L4,IA+2))
6200=      &   - DIST(IK,JK)
6210=      &   - DIST(SSTR(L4,IA+1),SSTR(L4,IA+2))
6220=       IF (MINS(L9) .LE. PENAL2) GO TO 1895
6230=       KTR(L9) = L4
6240=       MINS(L9) = PENAL2
6250=       PSMIN(L9) = SSTR(L4,IA+1)
6260=       QSMIN(L9) = SSTR(L4,IA+2)
6270=       IMIN2(L9) = IK
6280=       JMIN2(L9) = JK
6290= 1895  CONTINUE
6300= 1889  CONTINUE
6310= 1870  CONTINUE
6320=       RETURN
6330=       END
6340=*-------------------------------------------------------------------
```

```
6350=       SUBROUTINE RCP2KST(NSTLIE,NBST,SSTR,NSTR2,MINS,KTR,
6360=      &    PSMIN,GSMIN,IMIN2,JMIN2)
6370=*
6380=*        RECALCULER P2K DES SSTR RESTANTS AVEC LES AUTRES SSTR
6390=*
6400=       INTEGER SSTR(12,180), NSTR2(10),MINS(10), KTR(10),IMIN2(10)
6410=       INTEGER PSMIN(10), GSMIN(10), JMIN2(10)
6420=       INTEGER FROM,TO,F,D,P,R
6430=       INTEGER GENARC,DIST,DIST1J,DISTJ1,C
6440=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
6450=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
6460=       DIMENSION PCAND(1)
6470=       DIMENSION RTN(20),LPR(12),RTT(2)
6480=       COMMON /LIMITE/ MXM,MXN
6490=       COMMON // GENARC(33 000)
6500=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
6510=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
6520=      *            NCAND
6530=       DIMENSION IREVFL(4)
6540=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
6550=      *            ISDB,TEO
6560=       COMMON DIST1J(180),DISTJ1(180),NBVILLE
6570=       K2 = NSTLIE - 1
6580=       DO 200 L1=1,NBST
6590=       IF (SSTR(L1,1) .EQ. 0) GO TO 200
6600=       K2 = K2 + 1
6610=       NSTR2(K2) = L1
6620=       MINS(K2) = 9999
6630=       CALL RCP2KS(L1,K2,SSTR,KTR,MINS,PSMIN,GSMIN,IMIN2,JMIN2
6640=      &  ,NBST)
6650=   200 CONTINUE
6660=       RETURN
6670=       END
6680=*-------------------------------------------------------------
```

```
6690=       SUBROUTINE ELIM(TR,KS,MINI,NBST)
6700=       INTEGER TR(10), MINI(10)
6710=       K1 = 1
6720=    10 IF (TR(K1) .EQ. KS) GO TO 20
6730=       K1 = K1 + 1
6740=       GO TO 10
6750=    20  MINI(K1) = -9999
6760=       RETURN
6770=       END
6780=       SUBROUTINE CHCHIND(TR,KTRLIE,L9)
6790=       INTEGER TR(10)
6800=       K1 = 1
6810=    10 IF (TR(K1) .EQ. KTRLIE) GO TO 20
6820=       K1 = K1 + 1
6830=       GO TO 10
6840=    20  L9 = K1
6850=       RETURN
6860=       END
6870=*-------------------------------------------------------------
6880=       SUBROUTINE CHECKST(TOUR,SSTR,NBST,MI)
6890=*
6900=*            FORMER LES TOURS ET LES SS-TR.
6910=*
6920=       INTEGER FROM,TO,F,D,P,R
6930=       INTEGER GENARC,DIST,DIST1J,DISTJ1,C
6940=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
6950=       INTEGER  SOURCE, DESTIN,PLM, NARC
6960=       INTEGER CHECK(2,180), TOUR(42,180), SSTR(12,180)
6970=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
6980=       DIMENSION PCAND(1)
6990=       DIMENSION RTN(20),LPR(12),RTT(2)
7000=       COMMON /LIMITE/ MXM,MXN
7010=       COMMON // GENARC(33 000)
7020=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
7030=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
7040=      *          NCAND
7050=       DIMENSION IREVFL(4)
7060=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
7070=      *          ISDB,TEO
7080=       COMMON DIST1J(180),DISTJ1(180),NBVILLE
7090=       C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
7100=       FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
7110=       H(I)=SHIFT(GENARC(I),-50)
7120=       TO(I)=SHIFT(GENARC(I),-30).AND.1777B
7130=*
7140=*    ON COMMENCE PAR N TOURS
7150=*    -->  CHECK(1,I9) = 1
7160=*
7170=*    CHECK(2,I9) = 0  :    SOMMET NON MARQUE
7180=*
7190=*
7200=       DO 900 I9=1,NBVILLE
7210=       CHECK(1,I9) = 1
7220=       CHECK(2,I9) = 0
7230=   900 CONTINUE
7240=       DO 950 I=1,N
7250=       NCOUT = C(I)
7260=*       PRINT *," C(I) ",NCOUT
7270=       FLOT=-H(I)
7280=       IF(FLOT.GE.0)GOTO 920
7290=       FLOT=0
7300=       NOEUD=FROM(I)
7310=       IF(ARC(NOEUD).EQ.I)GOTO 910
7320=       NOEUD=TO(I)
7330=       IF(ARC(NOEUD).NE.I)GOTO 920
7340=   910 FLOT=X(NOEUD)
7350=   920 FLOT=FLOT
7360=*       PRINT *,"I,FROM(I),TO(I),FLOT",I,FROM(I),TO(I),FLOT
7370=*
7380=       IF(FLOT.LE.0)GOTO 950
7390=*       WRITE(9,*) "I ",I,"FROM(I) ",FROM(I),"TO(I) ",TO(I),"FLOT",FLOT,
7400=*      +   " U(I) ",U(I)
7410=       IF ((FROM(I) .EQ. 1) .OR. (FROM(I) .GT. NBVILLE)) GO TO 950
7420=*                    (SAUTER CET ARC)
```

```
7430=*
7440=*
7450=*    CHECK(1,COL)  :  CONTIENT L'ARC DE COL --> CHECK(1,COL)
7460=*
7470=       CHECK(1,FROM(I)) = TO(I) - NBVILLE
7480=*
7490=   950 CONTINUE
7500=*
7510=*      (COMPTER NB. DE COMPOSANTES)
7520=*
7530=*       PRINT *,(CHECK(1,I3),I3=1,NBVILLE)," CHECK "
7540=*       WRITE(9,*)(CHECK(1,I3),I3=1,NBVILLE)," CHECK "
7550=       NBST = 0
7560=       MI = 0
7570=       DO 1100 I9=2,NBVILLE
7580=       IF (CHECK(2,I9) .EQ. 1) GO TO 1100
7590=       NOANC = CHECK(1,I9)
7600=       MI = MI + 1
7610=       TOUR(MI,2) = 1
7620=       TOUR(MI,3) = I9
7630=       IND = 3
7640=       I2 = I9
7650=  1150  CHECK(2,I2) = 1
7660=       IF (CHECK(1,I2) .EQ. 1) GO TO 1130
7670=*       PRINT *," NOANC ",NOANC," CHECK(1,I2) ",CHECK(1,I2)
7680=  1180 I2 = CHECK(1,I2)
7690=       IND = IND + 1
7700=       TOUR(MI,IND) = I2
7710=*       PRINT *," I2 ",I2," IND ",IND
7720=       IF (CHECK(2,I2) .NE. 1) GO TO 1150
7730=       IF (CHECK(1,I2) .EQ. 1) GO TO 1160
7740=       IF (CHECK(1,I2) .NE. NOANC) GO TO 1160
7750=*
7760=*    ON A TROUVE LE SOUS-TOUR
7770=*
7780=*       PRINT *," TOUR SSTR ",(TOUR(MI,I4),I4=1,IND)
7790=       NBST = NBST + 1
7800=       SSTR(NBST,1) = IND - 3
7810=       IND = IND - 1
7820=       DO 1182 I6=2,IND
7830=         SSTR(NBST,I6) = TOUR(MI,I6+1)
7840=  1182 CONTINUE
7850=       MI = MI - 1
7860=*       LSTOUR = 0
7870=*       I7 = IND - 2
7880=*       DO 100 I6=1,I7
7890=*       LSTOUR = LSTOUR + DIST(SSTR(NBST,I6+1),SSTR(NBST,I6+2))
7900=*       PRINT *," LSTOUR ",LSTOUR
7910=*   100 CONTINUE
7920=*       PRINT *," LONG DE CETTE SSTR ",LSTOUR
7930=*       PRINT *," NBST",NBST," SSTR(NBST)",(SSTR(NBST,I6),I6=1,IND)
7940=       GO TO 1100
7950=  1130  IF (NOANC .EQ. 1) GO TO 1135
7960=       IND = IND + 1
7970=       TOUR(MI,IND) = 1
7980=       TOUR(MI,1) = IND - 2
7990=       GO TO 1100
8000=  1135  TOUR(MI,1) = 2
8010=       TOUR(MI,2) = 1
8020=       TOUR(MI,3) = I9
8030=       TOUR(MI,4) = 1
8040=       GO TO 1100
8050=  1160  IM = 0
8060=  1165  IM = IM + 1
8070=       IF (IM .GT. MI) GO TO 1195
8080=       IF (TOUR(IM,3) .NE. TOUR(MI,IND)) GO TO 1165
```

```
8090=*
8100=*      LIER DEUX PARTIES DU TOUR
8110=*
8120=       IND1 = TOUR(IM,1) + IND - 1
8130=       IND2 = TOUR(IM,1) + 2
8140=       TOUR(IM,1) = IND1 - 2
8150= 1166   TOUR(IM,IND1) = TOUR(IM,IND2)
8160=       IND2 = IND2 - 1
8170=       IND1 = IND1 - 1
8180=       IF (IND2 .NE. 2) GO TO 1166
8190=       IND = IND - 1
8200= 1168  TOUR(IM,IND1) = TOUR(MI,IND)
8210=       IND1 = IND1 - 1
8220=       IND = IND - 1
8230=       IF (IND .NE. 2) GO TO 1168
8240=       MI = MI - 1
8250=       IND = TOUR(IM,1) + 2
8260=       GO TO 1100
8270= 1195   TOUR(MI,1) = IND - 2
8280= 1100  CONTINUE
8290=*       DO 1500 I4=1,MI
8300=*       LIM = TOUR(I4,1) + 2
8310=*       LTOUR = 0
8320=*       DO 110 I5=1,LIM
8330=*       LTOUR = LTOUR + DIST(TOUR(I4,I5+1),TOUR(I4,I5+2))
8340=*       PRINT *," LTOUR ",LTOUR
8350=*   110 CONTINUE
8360=*       PRINT *," LONG. DE CE TOUR ",LTOUR
8370=**       WRITE(9,*) " TOUR ",I4," = ",(TOUR(I4,I5),I5=1,LIM)
8380=*       PRINT *,"1500 TOUR ",(TOUR(I4,I5),I5=1,LIM)
8390=* 1500 CONTINUE
8400=*       WRITE(9,*) " NBSSTOUR  = " ,NBST
8410=       PRINT *," NBST ",NBST
8420=       RETURN
8430=       END
8440=       SUBROUTINE COPIE(T1,T2,INDICE)
8450=       INTEGER T1(42,180), T2(42,180)
8460=       DO 10 I4=1,INDICE
8470=       NTAMP = T1(I4,1) + 2
8480=       DO 20 I5=1,NTAMP
8490=       T2(I4,I5) = T1(I4,I5)
8500=    20  CONTINUE
8510=    10  CONTINUE
8520=       RETURN
8530=       END
8540=*-------------------------------------------------------------------
8550=       SUBROUTINE COPIE2(T1,T2,INDICE)
8560=       INTEGER T1(12,180), T2(12,180)
8570=       DO 10 I4=1,INDICE
8580=       NTAMP = T1(I4,1) + 2
8590=       DO 20 I5=1,NTAMP
8600=       T2(I4,I5) = T1(I4,I5)
8610=    20  CONTINUE
8620=    10  CONTINUE
8630=       RETURN
8640=       END
8650=*-------------------------------------------------------------------
```

```
8660=       SUBROUTINE HRISTIQ(NBSTR,NBM,TR,STR,LONGSUP)
8670=*
8680=*          HEURISTIQUE POUR LIER  LES SOUS-TOURS AVEC LES TOURS
8690=*          D'APRES LE PRINCIPE D'OPTIMALITE
8700=*
8710=       INTEGER FROM,TO,F,D,P,R
8720=       INTEGER GENARC,DIST,DIST1J,DISTJ1,C
8730=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
8740=       INTEGER   PENAL2,TR(42,180),  STR(12,180)
8750=       INTEGER NBST,  STR2(42,180),  MI,  TOUR(42,180),  SSTR(12,180)
8760=       INTEGER NSTR(10),  IMIN1(10),JMIN1(10)
8770=       INTEGER NTR(10),  MIN2(10),PMIN(10),QMIN(10)
8771=       INTEGER VCORR(111)
8780=       INTEGER KTR(10),  MINS(10),  PSMIN(10),  QSMIN(10)
8790=       INTEGER NSTR2(10),  IMIN2(10),  JMIN2(10)
8800=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
8810=       DIMENSION PCAND(1)
8820=       DIMENSION RTN(20),LPR(12),RTT(2)
8830=       COMMON /LIMITE/ MXM,MXN
8840=       COMMON // GENARC(33 000)
8850=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
8860=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
8870=      *        NCAND
8880=       DIMENSION IREVFL(4)
8890=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
8900=      *         ISDB,TEO
8910=       COMMON DIST1J(180),DISTJ1(180),NBVILLE
8920=       C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
8930=       FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
8940=       H(I)=SHIFT(GENARC(I),-50)
8950=       TO(I)=SHIFT(GENARC(I),-30).AND.1777B
8960=       CALL COPIE(TR,TOUR,NBM)
8970=       CALL COPIE2(STR,SSTR,NBSTR)
8980=       LZ1 = LONGSUP
8990=       MI = NBM
9000=       NBST = NBSTR
9010=       CALL SECOND(T1)
9020=       NSTLIE = 1
9030=*
9040=*    CALCULER P1K
9050=*
9060=         KS = 1
9070= 1700    IF (KS .GT. NBST) GO TO 1910
9080=*
9090=*        NST(10) : INDICE DE SOUS-TOUR
9100=*        NASSTR  : NBR D'ARCS DE SOUS-TOUR
9110=*        NATR    : NBR. D'ARCS DU TOUR
9120=*
9130=         NSTR(KS) = KS
9140=         MIN2(KS) = 9999
9150=         NASSTR = SSTR(KS,1) + 1
9160=         DO 1720 I2=2,NASSTR
9170=         IK = SSTR(KS,I2)
9180=         JK = SSTR(KS,I2 + 1)
9190=         DO 1800 IT3=1,MI
9200=         NATR = TOUR(IT3,1)
9210=         DO 1900 IA=1,NATR
9220=         PENAL2 = DIST(TOUR(IT3,IA+1),JK)
9230=        &   + DIST(IK,TOUR(IT3,IA+2))
9240=        &   - DIST(IK,JK)
9250=        &   - DIST(TOUR(IT3,IA+1),TOUR(IT3,IA+2))
9260=         IF (MIN2(KS) .LE. PENAL2) GO TO 1900
9270=         MIN2(KS) = PENAL2
9280=         PMIN(KS) = TOUR(IT3,IA+1)
9290=         QMIN(KS) = TOUR(IT3,IA+2)
9300=         IMIN1(KS) = IK
9310=         JMIN1(KS) = JK
9320=         NTR(KS) = IT3
9330=*         PRINT *," PENAL2 ",PENAL2," PMIN ",PMIN(KS)," QMIN ",QMIN(KS),
9340=*        &    " IK ",IK," JK ",JK," IT3 ",IT3
9350= 1900    CONTINUE
9360= 1800    CONTINUE
```

```
9370= 1720 CONTINUE
9380=      KS = KS + 1
9390=      GO TO 1700
9400= 1910 IF (NBST .LE. 1) GO TO 2900
9410=      CALL TRIER(MIN2,PMIN,QMIN,NBST,NTR,IMIN1,JMIN1,NSTR)
9420=*      PRINT *,"MI2 ",(MIN2(I4),I4=1,NBST)," PM",(PMIN(I4),I4=1,NBST)
9430=*     &  ," QM ",(QMIN(I4),I4=1,NBST)," NST",(NSTR(I4),I4=1,NBST)
9440=*      PRINT *," IMIN1 ",(IMIN1(I4),I4=1,NBST),"JMIN1 ",
9450=*     &  (JMIN1(I4),I4=1,NBST),"NTR ",(NTR(I4),I4=1,NBST)
9460=*
9470=*     CALCULER P2K AVEC LES AUTRES SSTOUR .
9480=*
9490=      IF (NBST .LE. 1) GO TO 2900
9500=      DO 1850 L1=1,NBST
9510=      NSTR2(L1) = L1
9520=      MINS(L1) = 9999
9530=      NASSTR = SSTR(L1,1) + 1
9540=*      PRINT *," NASSTR ",NASSTR
9550=      DO 1870 L2=2,NASSTR
9560=      IK = SSTR(L1,L2)
9570=      JK = SSTR(L1,L2+1)
9580=      DO 1889 L4=1,NBST
9590=      IF (L4 .EQ. L1) GO TO 1889
9600=      NAAST = SSTR(L4,1)
9610=*      PRINT *," L4 ",L4,"L1 ",L1," IK",IK," JK ",JK," NA ",NAAST
9620=      DO 1895 IA=1,NAAST
9630=      PENAL2 = DIST(SSTR(L4,IA+1),JK)
9640=     &  + DIST(IK,SSTR(L4,IA+2))
9650=     &  - DIST(IK,JK)
9660=     &  - DIST(SSTR(L4,IA+1),SSTR(L4,IA+2))
9670=      IF (MINS(L1) .LE. PENAL2) GO TO 1895
9680=      KTR(L1) = L4
9690=      MINS(L1) = PENAL2
9700=      PSMIN(L1) = SSTR(L4,IA+1)
9710=      QSMIN(L1) = SSTR(L4,IA+2)
9720=      IMIN2(L1) = IK
9730=      JMIN2(L1) = JK
9740= 1895 CONTINUE
9750=*      PRINT *," PENAL2 ",PENAL2," L4 ",L4," PSMIN(L1)",PSMIN(L1)
9760=*     &  ,"QS  ",QSMIN(L1)," IK ",IK," JK ",JK
9770= 1889 CONTINUE
9780= 1870 CONTINUE
9790= 1850 CONTINUE
9800=      CALL TRIER(MINS,PSMIN,QSMIN,NBST,KTR,IMIN2,JMIN2,NSTR2)
9810=*      PRINT *,"MINS ",(MINS(I4),I4=1,NBST),(PSMIN(I4),I4=1,NBST)
9820=*     &  ," QS ",(QSMIN(I4),I4=1,NBST),"KTR ",(KTR(I4),I4=1,NBST)
9830=*      PRINT *," IMIN2 ",(IMIN2(I4),I4=1,NBST)," JMIN2  ",
9840=*     &  (JMIN2(I4),I4=1,NBST)," NSTR2 ",(NSTR2(I4),I4=1,NBST)
9850=*
9860=*            CHERCHER KT : TOUR A LIER ET
9870=*                      KS : SOUS-TOUR A LIER
9880=*
9890= 1890 IF (MINS(NSTLIE) .GE. MIN2(NSTLIE)) GO TO 2900
9900=*
9910=*      LIER SOUS-TOUR AVEC SOUS-TOUR
9920=*
9930=*      PRINT *," LIER SS-TR AVEEC S-TR "
9940=      WRITE(9,*)" LIER SOUS-TOUR AVEC SOUS-TOUR "
9950=      DO 1801 K9=1,NBST
9960=      K8 = SSTR(K9,1) + 2
9970=      DO 1802 K7=1,K8
9980=      STR2(K9,K7) = SSTR(K9,K7)
9990= 1802  CONTINUE
10000= 1801 CONTINUE
10010=      KTRLIE = KTR(NSTLIE)
10020=      KS = NSTR2(NSTLIE)
10030=      IK = IMIN2(NSTLIE)
10040=      NQ = QSMIN(NSTLIE)
10050=*      PRINT *," LIER SSTR ",KS," DANS SSTR ",KTRLIE
```

```
10060=       CALL LIAISON(KTRLIE,STR2,SSTR,KS,NG,IK)
10070=       LONGSUP = LONGSUP + MINS(NSTLIE)
10080=       DO 1901 K9=1,NBST
10090=       K8=STR2(K9,1) + 2
10100=       DO 1902 K7=1,K8
10110=       SSTR(K9,K7) = STR2(K9,K7)
10120= 1902 CONTINUE
10130= 1901 CONTINUE
10140=*       PRINT *," SSTR "
10150=*       DO 1940 I4=1,NBST
10160=*       LIM = SSTR(I4,1) + 2
10170=*       PRINT *," 1940 SSTR ",(SSTR(I4,I5),I5=1,LIM)
10180=* 1940 CONTINUE
10190=       SSTR(KS,1) = 0
10200=       CALL ELIM(NSTR,KS,MIN2,NBST)
10210=*       PRINT *," MIN2 ",(MIN2(I4),I4=1,NBST)
10220=       CALL CHCHIND(NSTR,KTRLIE,KL)
10230=*       PRINT *," KL ",KL
10240=       CALL RCP2KT(KTRLIE,KL,TOUR,SSTR,MIN2,PMIN,GMIN,IMIN1,JMIN1,
10250=      &      NTR,MI)
10260=*       PRINT *,"MIN2 ",(MIN2(I4),I4=1,NBST),(PMIN(I4),I4=1,NBST)
10270=*      &  ," GMIN ",(GMIN(I4),I4=1,NBST),(NTR(I4),I4=1,NBST)
10280=*       PRINT *," IMIN1 ",(IMIN1(I4),I4=1,NBST),"JMIN1 ",
10290=*      &  (JMIN1(I4),I4=1,NBST),"NST ",(NSTR(I4),I4=1,NBST)
10300=       NSTLIE = NSTLIE + 1
10310=       CALL TRIER(MIN2,PMIN,GMIN,NBST,NTR,IMIN1,JMIN1,NSTR)
10320=       IF (NSTLIE .EQ. NBST) GO TO 2900
10330=       CALL ELIM(NSTR2,KS,MINS,NBST)
10340=*       PRINT *," MINS ",(MINS(I4),I4=1,NBST)
10350=       CALL CHCHIND(NSTR2,KTRLIE,L9)
10360=*       PRINT *," L9 ",L9
10370=       DO 1960 K1=1,NBST
10380=       IF (KTR(K1) .EQ. KS) MINS(K1) = 9999
10390= 1960 CONTINUE
10400=       CALL RCP2KS(KTRLIE,L9,SSTR,KTR,MINS,PSMIN,QSMIN,IMIN2,JMIN2,
10410=      &  NBST)
10420=*       PRINT *,"MINS ",(MINS(I4),I4=1,NBST),(PSMIN(I4),I4=1,NBST)
10430=*      &  ," QS ",(QSMIN(I4),I4=1,NBST)," KTR ",(KTR(I4),I4=1,NBST)
10440=*       PRINT *," IMIN2 ",(IMIN2(I4),I4=1,NBST)," JMIN2 ",
10450=*      &  (JMIN2(I4),I4=1,NBST)," NSTR2 ",(NSTR2(I4),I4=1,NBST)
10460=       CALL TRIER(MINS,PSMIN,QSMIN,NBST,KTR,IMIN2,JMIN2,NSTR2)
10470=       GO TO 1890
10480=*
10490=*                  LIER AVEC UN TOUR
10500=*
10510=*          TPG  :  NUMERO DU TOUR A LIER
10520=*
10530=*
10540= 2900   KS = NSTR(NSTLIE)
10550=       IK = IMIN1(NSTLIE)
10560=       NTRLIE = NTR(NSTLIE)
10570=       NG = GMIN(NSTLIE)
10580=*       WRITE(9,*)" LIER SSTR ",KS," DANS TOUR ",NTRLIE
10590=*       PRINT *," LIER SOUS TOUR ",KS,"DANS ",NTRLIE
10600=       CALL LIAISON(NTRLIE,TOUR,SSTR,KS,NG,IK)
10610=*       PRINT *," FIN DE LIAISON 10440 "
10620=       LONGSUP = LONGSUP + MIN2(NSTLIE)
10630=       SSTR(KS,1) = 0
10640=       IF (NBST - NSTLIE .EQ. 0) GO TO 4000
10650=       MIN2(NSTLIE) = -9999
10660=       ITLIE = NTR(NSTLIE)
10670=       K2 = NSTLIE
10680=       NSTLIE = NSTLIE + 1
10690=       DO 2910 K1=1,NBST
10700=       IF (SSTR(K1,1) .EQ. 0) GO TO 2910
10710=       K2 = K2 + 1
10720=       IF (NTR(K2) .EQ. ITLIE) MIN2(K2) = 9999
10730=       NATR = TOUR(ITLIE,1)
10740=       NASSTR = SSTR(K1,1) + 1
```

```
10750=*       PRINT *," NASSTR ",NASSTR," K1 ",K1," ITLIE ",ITLIE
10760=*      &  ," K2 ",K2," NATR ",NATR
10770=       DO 2920 I2=2,NASSTR
10780=       IK = SSTR(K1,I2)
10790=       JK = SSTR(K1,I2+1)
10800=       DO 2930 IA=1,NATR
10810=       PENAL2 = DIST(TOUR(ITLIE,IA+1),JK)
10820=      &  + DIST(IK,TOUR(ITLIE,IA+2))
10830=      &  - DIST(IK,JK)
10840=      &  - DIST(TOUR(ITLIE,IA+1),TOUR(ITLIE,IA+2))
10850=       IF (MIN2(K2) .LE. PENAL2) GO TO 2930.
10860=       NSTR(K2) = K1
10870=       MIN2(K2) = PENAL2
10880=       PMIN(K2) = TOUR(ITLIE,IA+1)
10890=       GMIN(K2) = TOUR(ITLIE,IA+2)
10900=       IMIN1(K2) = IK
10910=       JMIN1(K2) = JK
10920=       NTR(K2) = ITLIE
10930=*       PRINT *," PENAL2 ",PENAL2," PMIN ",PMIN(K2)," GMIN ",GMIN(K2),
10940=*      &    " IK ",IK," JK ",JK," ITLIE ",ITLIE
10950= 2930 CONTINUE
10960= 2920 CONTINUE
10970= 2910 CONTINUE
10980=*       PRINT *,"MIN2 ",(MIN2(I4),I4=1,NBST),(PMIN(I4),I4=1,NBST)
10990=*      &  ," GMIN ",(GMIN(I4),I4=1,NBST)
11000=       IF (NBST - NSTLIE .LE. 0) GO TO 2900
11010=       CALL TRIER(MIN2,PMIN,GMIN,NBST,NTR,IMIN1,JMIN1,NSTR)
11020=*       PRINT *,"MIN2 ",(MIN2(I4),I4=1,NBST),(PMIN(I4),I4=1,NBST)
11030=*      &  ," GMIN ",(GMIN(I4),I4=1,NBST),(NTR(I4),I4=1,NBST)
11040=       CALL RCP2KST(NSTLIE,NBST,SSTR,NSTR2,MINS,KTR,PSMIN,
11050=      &  GSMIN,IMIN2,JMIN2)
11060=*       PRINT *," IMIN2 ",(IMIN2(I4),I4=1,NBST)," JMIN2  ",
11070=*      &  (JMIN2(I4),I4=1,NBST)," NSTR2 ",(NSTR2(I4),I4=1,NBST)
11080=       CALL TRIER(MINS,PSMIN,GSMIN,NBST,KTR,IMIN2,JMIN2,NSTR2)
11090=*       PRINT *,"MINS ",(MINS(I4),I4=1,NBST),(PSMIN(I4),I4=1,NBST)
11100=*      &  ," GS ",(GSMIN(I4),I4=1,NBST)," KTR ",(KTR(I4),I4=1,NBST)
11110=       GO TO 1890
11120= 4000 CALL SECOND(T2)
11130=       PRINT *," LONG T ",LONGSUP
11140=      & ," TEMPS DE HEURIST.  ",T2 - T1
11150=       LONGDIF = 0
11160=       AMELIO = FLOAT(LONGSUP - LZ1) / FLOAT(LONGSUP)
11170=       LONGDIF = LONGSUP - LZ1
11180=       PRINT *," DIFF ",LONGDIF," % D'ELOIGNEMENT ",AMELIO
11181=*       DO 4100 I=1,NBM
11182=*       L1 = TOUR(I,1)
11183=*       VCORR((I-1) * N + 1) = TOUR(I,3)
11184=*       DO 4200 I2=3,L1
11185=*       VCORR(TOUR(I,I1)) = TOUR(I,I1+1)
11186=* 4200 CONTINUE
11187=*       VCORR(TOUR(I,I1+1)) = NBVILLE + 1
11188=* 4100 CONTINUE
11189=*       VCORR(TOUR(I,I1+1)) = 1
11190=*       CALL SECOND(D1)
11191=*       CALL DEUXOPT(VCORR,TOTCORR,NBM)
11192=*       CALL SECOND(D2)
11193=*       PRINT *," TEMPS DE DEUXOPT ",D2 - D1
11194=*       PRINT *," TOTCORR ",TOTCORR
11195=*
11200=*    COMPTER LE NBR. D'ARCS DONT LEURS COUTS RELATIF >=
11210=*    ZHEURIST - ZOPTIMAL
11220=*
```

```
11230=*       NBRARC = 0
11240=*       DO 500 I=1,N
11250=*       FLOT = -H(I)
11260=*       IF (FLOT .GE. 0) GO TO 520
11270=*       FLOT = 0
11280=*       NOEUD = FROM(I)
11290=*       IF (ARC(NOEUD) .EQ. I) GO TO 510
11300=*       NOEUD = TO(I)
11310=*       IF (ARC(NOEUD) .NE. I) GO TO 520
11320=*   510 FLOT = X(NOEUD)
11330=*   520 FLOT = FLOT
11340=*       IF (FLOT .GT. 0) GO TO 500
11350=*       KOUT = C(I) - U(FROM(I)) + U(TO(I))
11360=*       IF ( KOUT .GE. LONGDIF) NBRARC = NBRARC + 1
11370=*   500 CONTINUE
11380=*       NBRBASE = 0
11390=*       DO 600 I=1,M
11400=*       NOARC = ARC(I)
11410=*       FLOT = X(I)
11420=*       IF (FLOT .GT. 0) GO TO 600
11430=*       KOUT = C(NOARC) - U(FROM(NOARC)) + U(TO(NOARC))
11440=*       IF (KOUT .GE. LDIF) NBRBASE = NBRBASE + 1
11450=*   600 CONTINUE
11460=*       NBRARC = NBRARC - NBRBASE
11470=*       PRINT *," NBR D'ARCS C.R. >= ZHEURIS - ZOPT", NBRARC
11480=*      &  ," % EST ",FLOAT(NBRARC) / FLOAT(N)
11490=        RETURN
11500=        END
11510=        SUBROUTINE AJARC(NOEUD1,NOEUD2,COUT,CAPAC)
11520=*
11530=*         CE MODULE AJOUTE UN ARC AU RESEAU EN FORMATION
11540=*
11550=*         PARAMETRES D'ENTREE:
11560=*
11570=*         - CAPAC :  CAPACITE DE L'ARC
11580=*         - COUT  :  COUT DE L'ARC
11590=*         - NOEUD1:  ORIGINE DE L'ARC
11600=*         - NOEUD2:  DESTINATION DE L'ARC
11610=*
11620=*         IMPLICIT INTEGER(A-W)
11630=        INTEGER FROM,TO,F,D,P,R
11640=        INTEGER COUT,CAPAC
11650=        INTEGER GENARC,C
11660=        INTEGER   BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
11670=        DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
11680=        DIMENSION PCAND(1)
11690=        DIMENSION RTN(20),LPR(12),RTT(2)
11700=        COMMON /LIMITE/ MXM,MXN
11710=        COMMON // GENARC(33 000)
11720=        COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
11730=        COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,
11740=       *          RTT,NCAND
11750=        DIMENSION IREVFL(4)
11760=        COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
11770=       *          ISDB,TEO
11780=*         COMMON DIST1J(180),DISTJ1(180),NBVILLE
11790=        C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
11800=        FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
11810=        H(I)=SHIFT(GENARC(I),-50)
11820=        TO(I)=SHIFT(GENARC(I),-30).AND.1777B
11830=*       PRINT *," NOEUD1,NOEUD2,COUT,CAPAC ",NOEUD1,NOEUD2,COUT,CAPAC
11840=*       &  ,"  N,",N," MXN ",MXN
11850=        N = N + 1
11860=        IF(N.GT.MXN) RETURN
11870=*       FROM(N)=NOEUD1
11880=*       TO(N)=NOEUD2
11890=*       C(N)=COUT
11900=*       H(N)=CAPAC
11910=*       PRINT *,"NOEUD1 NOEUD2 COUT ",NOEUD1,NOEUD2,COUT
11920=*         IF(COUT.GT.3777777777B) COUT=3777777777B
11930=*         IF(COUT.LT.-3777777777B) COUT=-3777777777B
11940=        GENARC(N)=SHIFT(CAPAC,50).OR.SHIFT(NOEUD1,40).OR.SHIFT(NOEUD2,30
11950=       +          .OR.COUT.AND.7777777777B
11960=        RETURN
11970=        END
```

```
11980=*-------------------------------------------------------------
11990=       SUBROUTINE CMPPPNA(STR,NBSTR,NUMST,NOMBARC)
12000=*
12010=*      CHERCHER LA COMPOSANTE AYANT LE PLUS PETIT NOMBRE D'ARCS
12020=*
12030=       INTEGER STR(12,180),  TNOARC(180),CNOARC(180)
12040=       INTEGER FROM,TO,F,D,P,R
12050=       INTEGER COUT,CAPAC
12060=       INTEGER GENARC,C
12070=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
12080=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
12090=       DIMENSION PCAND(1)
12100=       DIMENSION RTN(20),LPR(12),RTT(2)
12110=       COMMON /LIMITE/ MXM,MXN
12120=       COMMON // GENARC(33 000)
12130=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
12140=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRG,PO,PBAR,CP,PAS,RTN,LPR,
12150=      *            RTT,NCAND
12160=       DIMENSION IREVFL(4)
12170=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM,
12180=      *            ISDB,TEO
12190=*       COMMON DIST1J(180),DISTJ1(180),NBVILLE
12200=       C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
12210=       FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
12220=       H(I)=SHIFT(GENARC(I),-50)
12230=       TO(I)=SHIFT(GENARC(I),-30).AND.1777B
12240=       NOMBARC = 999
12250=       DO 10 I4=1,NBSTR
12260=*       PRINT *," STR(I,1) ",STR(I4,1)
12270=       CALL RECNOA(TNOARC,STR,I4,CNOARC)
12280=       NBA = STR(I4,1)
12290=       CALL COMPTER(NBRA,CNOARC,NBA)
12300=*       PRINT *," NBRA ",NBRA," SSTR ",I4," NBR ARCS ",STR(I4,1)
12310=       IF (NOMBARC .LT. NBRA) GO TO 10
12320=       NOMBARC = NBRA
12330=       NUMST = I4
12340=    10  CONTINUE
12350=*       PRINT *," SSTR CHOISI ",NUMST," NBRA ",NOMBARC
12360=       NOMBARC = STR(NUMST,1)
12370=       RETURN
12380=       END
12390=*-------------------------------------------------------------
12400=       SUBROUTINE COMPTER(NBRA,CNOARC,NBA)
12410=*        COMPTER LE NBR. D'ARCS NON IMPOSE A 1
12420=       INTEGER CNOARC(180)
12430=       NBRA = 0
12440=       DO 10 I3=1,NBA
12450=       IF (CNOARC(I3) .LE. -8000) GO TO 10
12460=       NBRA = NBRA + 1
12470=    10 CONTINUE
12480=       RETURN
12490=       END
```

```
12500=*-------------------------------------------------------------------
12510=       SUBROUTINE RECNOA(TNOARC,STR,NUMST,CNOARC)
12520=*
12530=*      RECHERCHE LES NUMEROS DES ARCS DE SS-TR "NUMST"
12540=*
12550=       INTEGER FROM,TO,F,D,P,R
12560=       INTEGER GENARC,DIST,DIST1J,DISTJ1,C
12570=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
12580=       INTEGER CNOARC(180),  TNOARC(180),  STR(12,180)
12590=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
12600=       DIMENSION PCAND(1)
12610=       DIMENSION RTN(20),LPR(12),RTT(2)
12620=       COMMON /LIMITE/ MXM,MXN
12630=       COMMON // GENARC(33 000)
12640=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
12650=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRG,PO,PBAR,CP,PAS,RTN,LPR,RTT,
12660=      *             NCAND
12670=       DIMENSION IREVFL(4)
12680=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM
12690=      *             ISDB,TEO
12700=*       COMMON DIST1J(180),DISTJ1(180),NBVILLE
12710=       C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
12720=       FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
12730=       H(I)=SHIFT(GENARC(I),-50)
12740=       TO(I)=SHIFT(GENARC(I),-30).AND.1777B
12750=       I4= 0
12760=       NBA = STR(NUMST,1)
12770=       DO 900 I2=1,NBA
12780=       I1=1
12790=   950  IF (I1 .GT. N) GO TO 900
12800=       FLOT = -H(I1)
12810=       IF(FLOT.GE.0)GOTO 920
12820=       FLOT=0
12830=       NOEUD=FROM(I1)
12840=       IF(ARC(NOEUD).EQ.I1)GOTO 910
12850=       NOEUD=TO(I1)
12860=       IF(ARC(NOEUD).NE.I1)GOTO 920
12870=   910 FLOT=X(NOEUD)
12880=   920 FLOT=FLOT
12890=       IF (FLOT .LE. 0) GO TO 945
12900=       IF (FROM(I1) .EQ. STR(NUMST,I2+1)) GO TO 960
12910=   945 I1 = I1 + 1
12920=       GO TO 950
12930=   960  I4 = I4 + 1
12940=       TNOARC(I4) = I1
12950=       CNOARC(I4) = C(I1)
12960=       I1 = N + 1
12970=       GO TO 950
12980=   900 CONTINUE
12990=       RETURN
13000=       END
13010=*-------------------------------------------------------------------
```

```
13020=       SUBROUTINE CRIAEA(IIA, IEA, IA, EA, NOMBARC, TNOARC, CNOARC)
13030=*
13040=*      CREER  LES  NOEUDS,  IA  ET  EA
13050=*
13060=       INTEGER TAMP, CNOARC(180), IA, EA, TNOARC(180)
13070=       DIMENSION IA(1000), EA(1000)
13080=       TAMP = IEA + 1
13090=       NENLEVE = 0
13100=       I3 = 1
13110=     3 IF (I3 .GT. NOMBARC) GO TO 15
13120=       IF (CNOARC(I3) .GT. -900) GO TO 5
13130=       I3 = I3 + 1
13140=       NENLEVE = NENLEVE + 1
13150=       GO TO 3
13160=     5 IEA = IEA + 1
13170=       EA(IEA) = TNOARC(I3)
13180=       I3 = I3 + 1
13190=       GO TO 3
13200=    15 NOMBARC = NOMBARC - NENLEVE
13210=       NTAMP2 = IEA - 1
13220=       DO 16 I5=TAMP, NTAMP2
13230=       IIA = IIA + 1
13240=       IA(IIA) = EA(I5)
13250=    16 CONTINUE
13260=       RETURN
13270=       END
13280=*-----------------------------------------------------------------------
13290=       SUBROUTINE ETAPE4(NIVEAU, NOMBARC, LONGINI, NZETOIL, EA, IA
13300=      &    , IEA, TABZ, NPROB, NUMARC, VALBASE, NPTIA1, NDESC, NBNDSG, NCFIXE,
13301=      &  NZCHOI, TTTRNET, NFOI)
13310=       INTEGER IA, NUMARC(1000), VALBASE, TABZ(1000), EA
13320=       DIMENSION IA(1000), EA(1000)
13330=       INTEGER TAMPZ, TAMPTR(42, 180), TAMPST(12, 180), TAMPNST, TAMPMI
13340=       INTEGER FROM, TO, F, D, P, R
13350=       INTEGER GENARCC
13360=       INTEGER    BT, PT, PAS, RTN, LPR, H, U, ARC, X, PCAND
13370=       DIMENSION P(380), F(380), D(380), U(380), ARC(380), X(380), R(380)
13380=       DIMENSION PCAND(1)
13390=       DIMENSION RTN(20), LPR(12), RTT(2)
13400=       COMMON /LIMITE/ MXM, MXN
13410=       COMMON // GENARC(33 000)
13420=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
13430=       COMMON /L/BT, MW, PT, M, NARC, IRT, MXIT, FRQ, PO, PBAR, CP, PAS, RTN, LPR, RT
13440=      *          NCAND
13450=       DIMENSION IREVFL(4)
13460=       COMMON /M/IREVFL, KARC, ICOM, ITH, MAXF, LNOD, MIC, MAC, NPE, IT, ITO, LGAM
13470=      *          ISDB, TEO
13480=*       COMMON DIST1J(180), DISTJ1(180), NBVILLE
13490=       C(I)=SHIFT(SHIFT(GENARC(I), 30), -30)
13500=       FROM(I)=SHIFT(GENARC(I), -40). AND. 1777B
13510=       H(I)=SHIFT(GENARC(I), -50)
13520=       TO(I)=SHIFT(GENARC(I), -30). AND. 1777B
13530=       PRINT *, "LONG", LONGINI, "ZETOI ", NZETOIL, "NIVEAU", NIVEAU, "NBA"
13540=      & , NOMBARC, " NZCHOI ", NZCHOI
13550=       I4=IEA - NOMBARC
13560=       IF (NIVEAU .EQ. 1) I4 = 0
13570=       DO 100 I2=1, NOMBARC
13580=       I4=I4 + 1
13590=       NPROB = NPROB + 1
13600=       TABZ(NPROB) = 9999
13610=       NUMARC(NPROB) = I2
13620=       CALL RNREV
13630=       IF (I2 .EQ. 1) GO TO 120
13640=*       PRINT *, " IA ", IA(I2 + NPTIA1 - 2)
13650=       CALL RNRVCJ(IA(I2 + NPTIA1-2), -900)
13660=*       PRINT *, " C(IA(I2 + NPTIA1 - 2))", C(IA(I2 + NPTIA1 -2))
13670=*       PRINT *, " I4 ", I4, "  EA(I4) ", EA(I4)
13680=   120  CONTINUE
13690=*       PRINT *, " EA(I4) ", EA(I4)
13700=       CALL RNRVCJ(EA(I4), 999)
13710=       CALL SECOND(TN1)
13720=       CALL RNREST
13730=       CALL SECOND(TN2)
13740=       PRINT *, " RTN(1) ", RTN(1), "  RTN(2)", RTN(2), " RTN(4) ",
13750=      &  RTN(4), "  RTN(5) ", RTN(5), " RTN(6) ", RTN(6)
13760=      &  , " TEMPS DE RNREST ", TN2 - TN1
13761=       TTTRNET = TTTRNET + (TN2 - TN1)
13762=       NFOI = NFOI + 1
13770=       NZG = LONGINI+RTN(2) + (900 * (I2-1)) + VALBASE + NCFIXE
13780=       PRINT *, " NZG ", NZG
13790=       IF (NZG .GE. NZETOIL) GO TO 140
13800=       NBNDSG = NBNDSG + 1
13810=*
```

```
13820=*-------------------------------------------------------------
13830=*       PARTIE AJOUTEE
13840=*       LDIF = NZETOIL - NZG
13850=*       NBRARC = 0
13860=*       DO 500 I=1,NARC
13870=*       FLOT = -H(I)
13880=*       IF (FLOT .GE. 0) GO TO 520
13890=*       FLOT = 0
13900=*       NOEUD = FROM(I)
13910=*       IF (ARC(NOEUD) .EQ. I) GO TO 510
13920=*       NOEUD = TO(I)
13930=*       IF (ARC(NOEUD) .NE. I) GO TO 520
13940=*   510 FLOT = X(NOEUD)
13950=*   520 FLOT = FLOT
13960=*       IF (FLOT .GT. 0) GO TO 500
13970=*       KOUT = C(I) - U(FROM(I)) + U(TO(I))
13980=**       PRINT *," FROM(I) ",FROM(I)," TO(I)",TO(I)," KOUT ",
13990=**      &  KOUT," FLOT ",FLOT
14000=*       IF ( KOUT .GE. LDIF) NBRARC = NBRARC + 1
14010=*   500 CONTINUE
14020=*       NBRBASE = 0
14030=*       DO 600 I=1,M
14040=*       NOARC = ARC(I)
14050=*       FLOT = X(I)
14060=*       IF (FLOT .GT. 0) GO TO 600
14070=*       KOUT = C(NOARC) - U(FROM(NOARC)) + U(TO(NOARC))
14080=*       IF (KOUT .GE. LDIF) NBRBASE = NBRBASE + 1
14090=*   600 CONTINUE
14100=*       NBRARC = NBRARC - NBRBASE
14110=*       PRINT *," NBR. D'ARCS = ",NBRARC," POURCENTAGE = ",
14120=*      &  FLOAT(NBRARC) / FLOAT(NARC)
14130=*-------------------------------------------------------------
14140=*
14150=       CALL CHECKST(TAMPTR,TAMPST,TAMPNST,TAMPMI)
14160=       IF (TAMPNST .EQ. 0) GO TO 130
14170=       IF (NZG .EQ. (NZETOIL - 1)) GO TO 135
14180=       TAMPZ = NZG
14190=       CALL HRISTIQ(TAMPNST,TAMPMI,TAMPTR,TAMPST,TAMPZ),
14200=       IF (TAMPZ .GE. NZETOIL) GO TO 135
14210=       NZETOIL = TAMPZ
14220=       PRINT *," ON A CHANGE NZETOIL "
14221=       IF (NZETOIL .EQ. NZCHOI) GO TO 900
14230=   135 TABZ(NPROB) = NZG
14240=       NPTG = NPTG + 1
14250=       NDESC = 1
14260=       GO TO 140
14270=   130  NZETOIL = NZG
14280=       PRINT *," ON A CHANGE NZETOIL "
14281=       IF (NZETOIL .EQ. NZCHOI) GO TO 900
14290=   140  CALL RNREV
14300=*       PRINT *," C(EA(I4))",C(EA(I4))," EA(I4) ",EA(I4)," I4 ",I4
14310=*
14320=*     REMETTRE L'ARC IMPOSE A 0 A SON COUT ORIGINAL
14330=*
14340=       CALL RNRVCJ(EA(I4),-999)
14350=   145 CALL RNREST
14360=*       PRINT *," RTN(1) ",RTN(1)," RTN(2) ",RTN(2)
14370=   100 CONTINUE
14380=*       PRINT *," NUMARC ",(NUMARC(I5),I5=1,NPROB)
14381=       GO TO 1000
14383=   900 NDESC = 0
14384=       I5= I2 + 1
14385=       IF (I5 .GT. NOMBARC) GO TO 1000
14386=       DO 910 I6=I5,NOMBARC
14387=       CALL RNRVCJ(IA(I6+NPTIA1-2),-900)
14388=   910 CONTINUE
14390=  1000 RETURN
14400=       END
```

```
14410=*-------------------------------------------------------------------
14420=       SUBROUTINE RPROBP(NPTIA1,NPTIA2,IA)
14430=*
14440=*        REMETTRE LES ARCS IMPOSES A 1 A LEURS COUTS ORIGINAL
14450=*
14460=       INTEGER FROM,TO,F,D,P,R
14470=       INTEGER GENARC,DIST,DIST1J,DISTJ1,C,IA
14480=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
14490=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
14500=       DIMENSION PCAND(1)
14510=       DIMENSION IA(1000)
14520=       DIMENSION RTN(20),LPR(12),RTT(2)
14530=       COMMON /LIMITE/ MXM,MXN
14540=       COMMON // GENARC(33 000)
14550=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
14560=       COMMON /L/BT,MW,PT,M,NARC,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,R1
14570=      *             NCAND
14580=       DIMENSION IREVFL(4)
14590=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAI
14600=      *             ISDB,TEO
14610=*       COMMON DIST1J(180),DISTJ1(180),NBVILLE
14620=       C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
14630=       FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
14640=       H(I)=SHIFT(GENARC(I),-50)
14650=       TO(I)=SHIFT(GENARC(I),-30).AND.1777B
14660=       PRINT *," REVENIR AU PROBLEME PERE  "
14670=       DO 10 I2=NPTIA1,NPTIA2
14680=       CALL RNRVCJ(IA(I2),900)
14690=*       PRINT *," IA(I2)  ",IA(I2),"C(IA(I2))",C(IA(I2))
14700=    10  CONTINUE
14710=       RETURN
14720=       END
14730=*-------------------------------------------------------------------
```

```
14740=       SUBROUTINE FIXER(LDIF,NCFIXE)
14750=*
14760=*        FIXER LES FLOTS DES ARCS HORS-BASE D'APRES LE COUT
14770=*        RELATIF.  SI !COUTS REL.! > ZSUP - ZINF ALORS
14780=*           CAS 1 : FLOT = 0 ; FIXER A 0
14790=*           CAS 2 : FLOT = 1 ; FIXER A 1
14800=*
14810=       INTEGER FROM,TO,F,D,P,R
14820=       INTEGER GENARC,C
14830=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
14840=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
14850=       DIMENSION PCAND(1)
14860=       DIMENSION RTN(20),LPR(12),RTT(2)
14870=       COMMON /LIMITE/ MXM,MXN
14880=       COMMON // GENARC(33 000)
14890=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
14900=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
14910=      *       NCAND
14920=        DIMENSION IREVFL(4)
14930=        COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAR
14940=      *       ISDB,TEO
14950=*       COMMON DIST1J(180),DISTJ1(180),NBVILLE
14960=        C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
14970=        FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
14980=        H(I)=SHIFT(GENARC(I),-50)
14990=        TO(I)=SHIFT(GENARC(I),-30).AND.1777B
15000=        PRINT *," LDIF ",LDIF
15010=        NCFIXE = 0
15011=        NZFO = 0
15020=        DO 500 L4=1,N
15030=        FLOT = -H(L4)
15040=        IF (FLOT .LT. 0) GO TO 490
15050=*         PRINT *," FROM ",FROM(L4)," TO ",TO(L4)
15060=        GO TO 495
15070=   490   FLOT = 0
15080=        NOEUD = FROM(L4)
15090=        IF (ARC(NOEUD) .EQ. L4) GO TO 500
15100=        NOEUD = TO(L4)
15110=        IF (ARC(NOEUD) .EQ. L4) GO TO 500
15120=*         PRINT *," FROM(I) ",FROM(L4)," TO(I) ",TO(L4)
15130=   495 KOUT = C(L4) - U(FROM(L4)) + U(TO(L4))
15140=*         PRINT *," FROM ",FROM(L4)," TO ",TO(L4)," FLOT ",FLOT,
15150=*       &   " KOUT ",KOUT," U(FR)",U(FROM(L4)),
15160=*       &   " U(TO) ",U(TO(L4))
15170=*         IF (FLOT .NE. 0) KOUT = - KOUT
15180=*         IF (KOUT .LE. LDIF) GO TO 500
15190=**          PRINT *," FROM ",FROM(L4)," TO ",TO(L4)," FLOT ",FLOT,
15200=**        &   " KOUT ",KOUT
15210=*         IF (FLOT .EQ. 0) GO TO 498
15211=        IF ((FLOT .EQ. 0) .AND. (KOUT .GE. LDIF)) GO TO 498
15212=        IF ((FLOT .EQ. 1) .AND. (KOUT .GE. LDIF)) GO TO 497
15213=*         PRINT *," FROM ",FROM(L4)," TO ",TO(L4)," FLOT ",FLOT,
15214=*       &   " KOUT ",KOUT," U(FR)",U(FROM(L4)),
15215=*       &   " U(TO) ",U(TO(L4))
15216=        GO TO 500
15220=   497 CALL RNRVCJ(L4,-999)
15230=        NCFIXE = NCFIXE + 1
15240=*         PRINT *," COUT ",C(L4)
15250=        GO TO 500
15251=   498   CONTINUE
15252=         PRINT *," FROM ",FROM(L4)," TO ",TO(L4)," FLOT ",FLOT,
15253=      &   " KOUT ",KOUT," U(FR)",U(FROM(L4)),
15254=      &   " U(TO) ",U(TO(L4))
15260=         CALL RNRVCJ(L4,900)
15261=        NZFO = NZFO + 1
15270=*         PRINT *," COUT ",C(L4)
15280=*         GO TO 500
15290=*  499    PRINT *," DANS BASE "
15300=*        KOUT = C(L4) - U(FROM(L4)) + U(TO(L4))
15310=*         PRINT *," FROM ",FROM(L4)," TO ",TO(L4),
15320=*       &   " KOUT ",KOUT," U(FR)",U(FROM(L4)),
15330=*       &   " U(TO) ",U(TO(L4))
15340=   500 CONTINUE
15341=        PRINT *," NBR. D'ARCS SONT FIXE A 1 ",NCFIXE
15342=        PRINT *," NBR. D'ARCS SONT FIXE A 0 ",NZFO
15343=        PRINT *," POURCENTAGE  =  ", FLOAT(NCFIXE +NZFO) /FLOAT(N)
15350=        RETURN
15360=        END
```

```
15370=*-------------------------------------------------------------
15380=       SUBROUTINE HEAP(TABSIJ,NARC)
15390=*
15400=*         TRIER LE TABLEAU TABSIJ, D'APRES LA METHODE HEAPSORT
15410=*
15420=       INTEGER TABSIJ(10000,2), NARC, LEFT, RIGHT, XFROMTO, XCOUT
15430=       LEFT = (NARC /2) + 1
15440=       RIGHT = NARC
15450=    10  IF (LEFT .EQ. 1) GO TO 20
15460=       LEFT = LEFT - 1
15470=       CALL SIFT(LEFT,RIGHT,TABSIJ)
15480=       GO TO 10
15490=    20  IF (RIGHT .EQ. 1) GO TO 30
15500=       XFROMTO = TABSIJ(1,1)
15510=       XCOUT = TABSIJ(1,2)
15520=       TABSIJ(1,1) = TABSIJ(RIGHT,1)
15530=       TABSIJ(1,2) = TABSIJ(RIGHT,2)
15540=       TABSIJ(RIGHT,1) = XFROMTO
15550=       TABSIJ(RIGHT,2) = XCOUT
15560=       RIGHT = RIGHT - 1
15570=       CALL SIFT(LEFT,RIGHT,TABSIJ)
15580=       GO TO 20
15590=    30  RETURN
15600=       END
15610=       SUBROUTINE SIFT(LEFT,RIGHT,TABSIJ)
15620=       INTEGER TABSIJ(10000,2),LEFT, RIGHT,XFROMTO,XCOUT
15630=       K1 = LEFT
15640=       K2 = 2 * K1
15650=       XFROMTO = TABSIJ(K1,1)
15660=       XCOUT = TABSIJ(K1,2)
15670=    10 IF (K2 .GT. RIGHT) GO TO 30
15680=       IF (K2 .GE. RIGHT) GO TO 20
15690=       IF (TABSIJ(K2,2) .LT. TABSIJ(K2+1,2)) K2 = K2 + 1
15700=    20 IF (XCOUT .GE. TABSIJ(K2,2)) GO TO 30
15710=       TABSIJ(K1,1) = TABSIJ(K2,1)
15720=       TABSIJ(K1,2) = TABSIJ(K2,2)
15730=       K1 = K2
15740=       K2 = 2 * K1
15750=       GO TO 10
15760=    30 TABSIJ(K1,1) = XFROMTO
15770=       TABSIJ(K1,2) = XCOUT
15780=       RETURN
15790=       END
15800=       INTEGER FUNCTION DIST(LIG,COL)
15810=       INTEGER LIG,COL
15820=       INTEGER FROM,TO,F,D,P,R
15830=       INTEGER GENARC,C
15840=       INTEGER DIST1J,DISTJ1
15850=       INTEGER    BT,PT,PAS,RTN,LPR,H,U,ARC,X,PCAND
15860=       DIMENSION P(380),F(380),D(380),U(380),ARC(380),X(380),R(380)
15870=       DIMENSION PCAND(1)
15880=       DIMENSION RTN(20),LPR(12),RTT(2)
15890=       COMMON /LIMITE/ MXM,MXN
15900=       COMMON // GENARC(33 000)
15910=       COMMON /E/P/F/F/G/D/H/U/I/ARC/J/X/K/R/N/PCAND
15920=       COMMON /L/BT,MW,PT,M,N,IRT,MXIT,FRQ,PO,PBAR,CP,PAS,RTN,LPR,RTT,
15930=      *          NCAND
15940=       DIMENSION IREVFL(4)
15950=       COMMON /M/IREVFL,KARC,ICOM,ITH,MAXF,LNOD,MIC,MAC,NPE,IT,ITO,LGAM
15960=      *          ISDB,TEO
15970=       COMMON DIST1J(180),DISTJ1(180),NBVILLE
15980=       C(I)=SHIFT(SHIFT(GENARC(I),30),-30)
15990=       FROM(I)=SHIFT(GENARC(I),-40).AND.1777B
16000=       H(I)=SHIFT(GENARC(I),-50)
16010=       TO(I)=SHIFT(GENARC(I),-30).AND.1777B
16020=       DIST = 999999
16030=       IF (LIG .EQ. COL) RETURN
16040=       IF (LIG .EQ. 1) GO TO 10
16050=       IF (COL .EQ. 1) GO TO 20
16060=       NOARC = ((LIG - 2) * (NBVILLE - 2)) + (COL - 2)
16070=       IF ((COL - 2) .LT. (LIG - 2)) NOARC = NOARC + 1
16080=*       PRINT *," DISTJ1(LIG)",DISTJ1(LIG)," DIST1J(COL) ",
16090=*      &    DIST1J(COL)
16100=       DIST = C(NOARC) + DISTJ1(LIG) + DIST1J(COL)
16110=       RETURN
16120=    10 DIST = DIST1J(COL)
16130=       RETURN
16140=    20 DIST = DISTJ1(LIG)
16150=       RETURN
16160=       END

16.15.07.  T1181AT.   1181-A0  EQ(02-PTX) PR      1     1667 LN     26 PG 86/12/1
```

REMERCIEMENTS

Je remercie toutes les personnes qui m'ont aidée dans la présentation de cette thèse. D'abord, j'exprime ma reconnaissance à Messieurs Gilbert Laporte et Yves Nobert du Centre de recherche sur les transports, pour leur précieuse aide. Mademoiselle Hélène Mercure et Monsieur Réjean Lessard m'ont patiemment expliqué les algorithmes PA et RNET.

Je suis reconnaissante aussi au Gouvernement du Québec pour l'aide financière qu'il m'a apportée au cours de mes études.

Finalement, je tiens à remercier Truong Nguyen qui a réussi à me motiver tout au long de cette étude.